Interface 1bis for the Sinclair ZX Spectrum 48k Ver 4b
Operating system reference

1. Compatibility

The 'Interface 1bis' is software compatible with the Sinclair 'ZX INTERFACE 1' at BASIC command as well as 'hook-code' level.

1.1 Necessary pre-conditions

- Same mechanism to extend the BASIC interpreter: paging a 'shadow' ROM in place of the BASIC ROM, whenever a syntax error is encountered
- Same 'extended BASIC' syntax
- Same system variables
- Same mechanism to access shadow ROM routines: 'Hook codes'
- Same data structure for handling sequential files: 'Microdrive channel'

1.2 Limitations

- BASIC commands and hook-codes referring to the RS-232 port and 'ZX NETWORK' are not implemented.

2. Memory layout

When activated, the interface disables the internal PROM of the ZX Spectrum and pages in its own operating system (OpSys), which resides in two contiguous 16 KB NVSRAM banks with the following layout:

Bank	Offset	Size	Address	Write protected
BASIC ROM	#0000	#4000	#0000	Yes
Shadow ROM	#4000	#2E00	#0000	Yes
Work RAM	#6E00	#0200	#2E00	No
Buffers	#7000	#1000	#3000	No

2.1 The 'BASIC ROM'

is a slightly modified copy of the ZX Spectrum 48k ROM, the scope of the changes being restricted to:

- enabling software-controlled memory paging by means of input/output operations to dedicated ports
- trapping the calls to the tape routines, to handle .TAP files,
- modified NMI handling, to allow the creation of (.Z80) snapshots,

and optionally:

- integration of an ESC/P printer driver in the ZX Spectrum BASIC
- fixing some known ZX Spectrum 48k ROM bugs.

2.2 The 'shadow ROM'

The shadow ROM is fully compatible with the 8 KB ROM of the original Sinclair 'ZX INTERFACE 1' at BASIC command and 'hook code' level,

- Following hook codes are not implemented:

Code	Function	Reason
#1D	R232 input	(1)
#1E	R232 output	(1)
#2D	Open network channel	(1)
#2E	Close network channel	(1)
#2F	Get packet	(1)
#30	Send packet	(1)
#34	Open "B" channel	(2)

(1) The corresponding hardware device is not supported
(2) Redundant

2.3 The 'work RAM'

- The 'work RAM' is structured as below:

Address	Block	Bytes
#2E00	Variables	128
#2E80	Internal stack	64
#2EC0	Printer buffer	64
#2F00	Page buffer	256

- The internal stack is used when handling (.Z80) snapshots or loading (.TAP) files.

2.4 Sector buffers

- There are seven sector buffers and one 'current directory table', of 512 bytes each.

3. The extended BASIC

3.1 Syntax

- All 'ZX INTERFACE 1' extended BASIC statements are accepted in their original format.
- A number of syntax enhancements are implemented.

```
3.1.1   CAT [#<str>;][fsq]
3.1.2   CLEAR #
3.1.3   CLOSE #<str>
3.1.4   CLS #
3.1.5   ERASE <fsq>|#<hdl>
3.1.6   FORMAT [#<csz>;][[{*}<dev>;]<dnr>]{;<nam>}
3.1.8   INKEY$ #<str>
3.1.9   INPUT #<str>,<var>
3.1.10  LOAD <fsp>|#<hdl>|STOP [<opt>]
3.1.11  MERGE <fsp>|#<hdl>
3.1.12  MOVE <fss>|#<sts> TO|AT <fsd>|#<std>
3.1.13  MOVE [<dvs>;]<dns>{;<nas>} OVER [<dvd>;]<dnd>{;<nad>}
3.1.14  MOVE #<str>|#<hdl> POINT [<pnt>]
3.1.15  OPEN #<str>;<fsq> [IN|OUT|RND]
3.1.16  PRINT #<str>;<exp>
3.1.17  SAVE <fsp>|#<hdl>|STOP [<opt>]
3.1.18  VERIFY <fsp>|#<hdl>|STOP [<opt>]

Where:  <str> = Stream (0-15)
        <sts> = Source stream number (0-15)
        <std> = Destination stream number (0-15)
        <hdl> = Handle (0-15)
        <fsq> = File specifier
              = [[{*}<dev>;]<dnr>];<nam>
        <fss> = Source file specifier
        <fsd> = Destination file specifier
        <fsp> = File specifier for SAVE, LOAD, VERIFY or MERGE
              = [[*<dev>;]<dnr>];<nam>
        <dev> = Device specifier
              = <typ>[<vol>]
        <dvs> = Source device specifier
        <dvd> = Destination device specifier
        <typ> = Device type literal
              = "V" - for:'serVer' drive
              = "M" - for: flash drive
        <vol> = Volume literal: "A"-"O"
        <dnr> = Drive number (1-255)
        <dns> = Source drive number (1-255)
        <dnd> = Destination drive number (1-255)
        <nam> = File name (1-10 characters)
        <nas> = Source file name (1-10 characters)
        <nad> = Destination file name (1-10 characters)
        <opt> = SAVE, LOAD or VERIFY options
              = LINE <lin>
              = DATA <ary>[$]()
              = CODE [<add>[,<len>{,<pnt>}]]
              = SCREEN$
              = BIN [<add>[,<len>[,<pnt>]]]
              = [;]<fty>
        <fty> = File type literal ( see 3.4 )
        <lin> = Auto-run line number (0-9999)
        <ary> = Array name
        <add> = Memory block address (0-65535)
        <len> = Memory block length  (0-65535)
        <pnt> = File pointer
              = <rec>[,<pos>]
        <rec> = Record number (0-32767)
        <pos> = Position within a record (0-511)
        <var> = BASIC variable
        <exp> = BASIC expression
        <csz> = Allocation unit in sectors/cluster (2,4,8,16)
```

- Syntax elements in square brackets are optional
- Syntax elements in curly brackets are accepted but not used
- Alternative syntax elements are separated by a vertical bar
- A (file) 'handle' is a stream opened to a file, using the option RND
- The position within a record: <pos> may be specified in the range (0-65535), because the pointer is always automatically normalized:

  <rec> = <rec>+int(<pos>/512)
  <pos> = mod(<pos>,512)
- The separator is required before the file type literal if the file name is specified by a string expression rather than a constant

3.2 Channels

3.2.1 The 'Microdrive' channel: M
- The M channel provides buffered character input/output from/to the supported storage devices
- It is compatible with the 'Microdrive' channel of the original 'ZX INTERFACE 1', having the same descriptor structure

3.2.2 The 'Local Area Network' channel: N
- Channel N is not implemented. Any reference to it produces an error report

3.2.3 The 'RS-232 Interface' channels: B and T
- These channels are implemented as output-only. Any input operation produces an error report
- Channel B sends binary data directly to the printer while channel T behaves identically to channel P

3.2.4 The 'Handle' channel: H
- An 'H channel' is created by opening a stream to a file, using the option RND. Its descriptor is identical to bytes 0-30 of the M channel descriptor
- Such a stream can be used as a 'handle' to specify the associated file in LOAD, SAVE, VERY, MERGE and ERASE statements

3.2.5 The 'NULL' channel: U
- Provides no input and discards any output

3.3 File names

- Full names may be composed of segments, separated by "/". The last segment represents the actual filename, while all the other make up the path. The length of each segment can not exceed 10 characters and the total length of the path 254 characters
- A name ending with a "/" represents a directory name
- Filenames may have a trailing 'file type literal', separated by a ".", as an extension
- A leading "/" stands for the root directory of the disk and a "../" for the parent directory
- For the 'server drive', "/A/", "/C/.."/Z/" represent the drives A, C .. Z of the server. The alternative form "a:/".. is also accepted.
- When not creating a new file, the wild cards "?" (standing for "any character") and "*" (standing for "any number of characters") are accepted in filenames, but never in directory names
- Filenames are case-insensitive

3.4 File types

3.4.1 'BASIC' files

Type	Literal	Description	Extension
0	P	BASIC program	ZZP
1	N	Number array	ZZN
2	A	String array	ZZA
3	C	CODE block	ZZC

- To allow access via the SAVE, LOAD and VERIFY commands these files contain a 9-byte header, with the following structure:

0	File type (0-3)
1-2	File length (excluding the header)
3-4	Loading address (Code)
5-6	Length of program only (Program) Array name (Numeric or String)
7-8	Start line (Program)

3.4.2 'Regular' files

Type	Literal	Description	Extension
4	F	PRINT file	ZZF

5	X	Text file	TXT
6		Spare	
7	B	Binary file	ZZB

- The maximum length of a regular file is 16 MByte (32768 records of 512 bytes each).

3.4.2.1 PRINT file (type 4)
- PRINT files are implemented as in the original ZX INTERFACE 1 extended BASIC, to be accessed via the OPEN#, PRINT#, INKEY$# and INPUT# commands

3.4.2.2 Text file (type 5)
- A 'Text' file is a PRINT file, with every CR followed by a LF and all BASIC tokens expanded
- When writing (PRINT#) to a stream opened to a 'text file', a LF is automatically inserted after each CR
- When reading (INPUT#) from a stream opened to a 'text file', any LF following a CR is discarded

3.4.2.3 Type number 6 is 'spare' (not implemented)

3.4.2.4 Binary file (Type 7)
- 'Binary' files have no specific structure

3.4.3 'Emulator' files

Type	Literal	Description	Extension
8	S	Screen dump	SCR
9		Spare	
10	T	'Tape' file	TAP
11	Z	'Z80' snapshot	Z80

3.4.3.1 Screen dump (type 8)
- A 'Screen dump' represents the contents of the video RAM, having the default loading address of: #4000 and the default length of: #1B00,
- Screen dumps are loaded or saved specifying the file type either by means of the filename extention '.s' or the command option 's'

3.4.3.2 Type number 9 is 'spare' (not implemented).

3.4.3.3 'Tape' file (type 10)
- A 'tape' file is opened for input or output via the LOAD or respectively SAVE statement, specifying the file type either by means of the filename extention '.t' or the command option 't', after which, all BASIC tape input or output is redirected to the specified file, until the end of the 'input tape' is reached, the length of the 'output tape' exceeds 16 MB or the file is closed using the LOAD or respectively SAVE command with the option: STOP
- A reset or even a power-off does not close the tape files.
- Opening the 'input tape' to a non-existing file will generate the error report "File not found"
- Opening the 'output tape' to a non-existing file will create that file
- Opening the 'output tape' to an existing file will append to that file
- The 'input tape' and 'output tape' can be simultaneously opened to the same file, but the blocks appended after the 'input tape' was opened, will not be accessible until the 'input tape' is closed and re-opened
- Opening the `input tape' using the file type 'T' (capital) will immediately perform the equivalent of NEW, followed by LOAD""

3.4.3.4 (.Z80) Snapshot file (type 11)
- (.Z80) snapshot files are launched using the LOAD command, specifying the file type either by means of the filename extention '.z' or the command option 'z'
- After loading a snapshot with the file type 'Z' (capital) the interface will switch to the 'ON - inactive' state
- To create a snapshot, a file must be first opened using the SAVE command specifying the file type either by means of the filename extention '.z' or the command option 'z', after which, generating a NMI saves the snapshot and closes the corresponding file
- Closing can also be forced using the VERIFY command with the option: STOP, but the resulting file will have no usable content

- A reset or even a power-off does not close the snapshot file
- All versions of 48k 'Z80' snapshots can be loaded, but only uncompressed version 1.45 snapshots can be created

3.4.4 Reserved file types

Type	Literal	Description	Extension
12		Reserved	
13		Reserved	
14		Reserved	
15	Y	Any type	*

- Type numbers: 12,13 and 14 are reserved
- Type number 15 is the 'type wild card', standing for "Any type"

3.4.5 Directories (type 16)

- Directories are special files, accessed via the commands LOAD, standing for 'change', SAVE, standing for 'create' and DELETE, containing sequences of 16-byte, fixed-length 'file specifiers':

Offset	Length	Description
0	1	File type
1	10	File name
11	2	First sector of the file
13	3	File length

with the byte #FF as an end marker.
- The size of a flash drive (sub)directory is limited only by the available space and the depth of the directory tree by the maximum length of the path name: 254 bytes.
- The first entry of the first record of a flash drive directory has the following structure:

Offset	Length	Description
0	1	Type: 16
1	10	Directory name
11	2	First sector of the parent directory, or 00 00 for the root directory
13	3	00 00 00

3.5 Other syntax issues

3.5.1 Default values

- The default values for the device literal <dev>, volume literal <vol> and drive number <dnr> are the ones last specified in a statement
- For statement 3.1.1, the default value of <str> is: 2
- For statement 3.1.6, the default value of <csz> is the one stored on the media when the drive was last formatted, or otherwise: 8

3.5.2 The CAT command

- The file list produced by the CAT statement has following layout:

Column		
	1-10	Filename
	12	File type literal
14-21		File length in bytes
23-27		Auto start line (Program)
		Array letter (Numeric or String)
		Loading address (Code)

- The number of free sectors available on the drive is given as the product of the number of free clusters and the cluster size
- If no name is specified, all files in the current directory are catalogued
- If a name is specified, then its path indicates the directory to be catalogued and the filename and extension are used as filters for the output of the command
- If followed by the token ABS, the CAT command outputs only the absolute path
- If followed by the token PEEK, the CAT command only sets the system variable SER_FL to a non-zero value if the specified file exists
- If the specified drive number is 0, then the name is considered a command and is sent to the peripheral port, to be interpreted by either the server, if it ends in a "/", or otherwise by the peripheral controller. After processing the command, these are expected to send a response, which is printed out as hex-dump, if not supressed by a NOT

option token

Following strings are available for configuring the mouse driver:

```
3.5.2.1 Enable mouse:           "[m]e[n]"
3.5.2.2 Disable mouse:          "[m]d[i]"
3.5.2.3 Mouse as joystick:      "[m]j[o]"
3.5.2.4 Legacy mode:            "[m]l[g]"
3.5.2.5 Windowed mode:          "[m]w[i]"
3.5.2.2 Set sensitivity:        "[m]s"+CHR$ <s>
        <s> = 3 - 32  Sensitivity of mouse as joystick
3.5.2.2 Set rate:               "[m]r"+CHR$ <r>
        <r> = 1 - 10  Mouse as joystick sample rate [Hz]
3.5.2.2 Set window width:       "[m]x"+CHR$ <x>
        <x> = 10 - 255  Window width in pixels
3.5.2.2 Set window height:      "[m]y"+CHR$ <y>
        <x> = 10 - 192  Window height in pixels
```

The command strings may be concatenated up to a length of 10 characters, in which case the characters in sqare brackets may be omitted if they are not the first or the last in the composite string.

3.5.3 The FORMAT command
- The statement 3.1.6 does not apply to the 'server' drive.
- The statement 3.1.6 with: <dnr> = 0 identifies a new device before its first use and also clears the 'current directories' table
- The allowed values of the cluster size in statement 3.1.6 are: 2, 4, 8 and 16. Any other number is disregarded and the default value used instead

3.5.4 The MOVE command
- If both source and destination are files, the operation is performed sector by sector, rather than byte by byte
- If source and destination are on the same drive and the separator AT is used, rather than TO, the source file is not copied but renamed
- The statement 3.1.12 is repetitive. It processes all files that match the specified source name.
- The statement 3.1.13, copies an entire logical drive
- The statement 3.1.14 sets the file pointer of the file, to which the stream is currently opened, to the specified position
If the stream is opened to a M-channel and the specified position is out of range, the file pointer is set to EOF

3.5.5 The OPEN command
- Any file can be opened for sequential access, not only PRINT files
- The optional keywords IN or OUT force the opening of the file for reading or respectively writing
- Opening a non-existing file with the option IN generates the error report "File not found"
- Opening an existing file with the option OUT sets the file pointer to EOF
- Opening a file with the option RND creates a random access 'handle' for that file

3.5.6 The SAVE, LOAD and VERIFY commands
- The option BIN allows to load, save or verify a memory block from/to a given position of any type of file
- If the file is accessed via a 'handle' rather than a specifier, the pointer entered with the option BIN is not used, but instead the one stored in the corresponding H channel descriptor, which is set to 0 when the file is opened and subsequently updated automatically following each operation.

3.5.7 The ERASE command
- The form: ERASE <fsq> is repetitive. It processes all files that match the specified name.
- The form: ERASE #<hdl> is not accepted for the 'server' drive.

3.5.8 The printer commands.
- The printer commands: LPRINT, LLIST and COPY work as expected with a ESC/P printer.
- The block graphics and UDG characters are printed as bitmaps at a density of 80 DPI.
- The system variables P_POSN and PR_CC are used as follows:

Variable	Address	Length	Description

P_POSN	#5C7F	23679	1	Column number
PR_CC	#5C80	23680	1	Lines per page minus Line number
	#5C81	23681	1	Bit 7 reset = 64 columns
				set = 32 columns
				Bits 0-6 = Lines per page

- OPEN #<str>,"P" sends an initialization string to the printer
- While the interface is connected to a server PC, the print jobs are forwarded to the server application, which directs them to a printer or a spool file.

3.6 Error messages

The error messages are the same as those of the original ZX INTERFACE 1, except for the following:

- 06: "Invalid station number" replaced by "Invalid path"
- 08: "Missing station number" replaced by "Feature not supported"
- 0A: "Missing baud rate" replaced by "Communication error"
- 0B: "Header mismatch error" replaced by "Directory in use"
- 13: "Hook code error" replaced by "File exists"
- 16: "Wrong file type" not used

4. Data structures

4.1 The ZX INTERFACE 1 system variables

Variable	Address		Length	Replaces
FLAGS3	#5CB6	23734	1	
VECTOR	#5CB7	23735	2	
..				
SER_FL	#5CC7	23751	2	
..				
CHADD_	#5CCB	23755	2	
..				
DRV_NR	#5CD6	23766	1	D_STR1
PTH_LN	#5CD7	23767	1	
STR_NR	#5CD8	23768	1	S_STR1
DEV_TY	#5CD9	23769	1	L_STR1
NAM_LN	#5CDA	23770	1	N_STR1
FIL_TY	#5CDB	23771	1	
NAM_AD	#5CDC	23772	2	P_STR1
DRV_N2	#5CDE	23774	1	D_STR2
PTH_L2	#5CDF	23775	1	
STR_N2	#5CE0	23776	1	S_STR2
DEV_T2	#5CE1	23777	1	L_STR2
NAM_L2	#5CE2	23778	1	N_STR2
FIL_T2	#5CE3	23779	1	
NAM_A2	#5CE4	23780	2	P_STR2
HD__00	#5CE6	23782	1	HD_00
HD__0B	#5CE7	23783	2	HD_0B
HD__0D	#5CE9	23785	2	HD_0D
HD__0F	#5CEB	23787	2	HD_0F
HD__11	#5CED	23789	1	HD_11
HD__DV	#5CEE	23790	1	
HD__DR	#5CEF	23791	1	COPIES

- The variables not shown are not used

4.1.1 FLAGS3

Bits 0-4 have the same significance as in the original ZX INTERFACE 1 'Shadow ROM'

- Bit 0 Shadow ROM entered the second time for the same error
- Bit 1 Shadow ROM entered the first time after creation of the new system variables, or
  CLEAR# command in progress
- Bit 2 Shadow ROM entered by means of a hook-code
- Bit 3 H(andle) channel SAVE / LOAD / VERIFY in progress, or
  CAT command in progress
- Bit 4 Character by character MOVE command in progress, or
  Destination name in MOVE command contains wild cards, or
  A filename was specified in the CAT command, or
  Suppress auto-run of a loaded BASIC program, or
  SAVE / LOAD option specified in upper case
- Bit 5 Find the 'last match' in a search operation
- Bit 6 Find the 'next match' in a search operation
- Bit 7 Temporary file opened on the 'server' drive

4.1.2 VECTOR, CHADD_

Same as in the original ZX INTERFACE 1 'Shadow ROM'

4.1.3 File specifiers

The two 8-byte file specifiers at DSTR_1 and DSTR_2 have the same function as in the original ZX INTERFACE 1 'Shadow ROM', except for the drive number's high byte, which is used to store the path name's length and the file name's length high byte, which is used to store the file type

4.1.4 BASIC header: HD__00 .. HD__11

Same as in the original ZX INTERFACE 1 'Shadow ROM'

4.1.5 HD__DV and HD__DR

Replace HD_11 high byte and COPIES. Store the device and drive number

4.2 The M channel descriptor

Offset	Name	Description

```
---------------------------------------------------------------------
  0               Address of error handling routine (0008)
  2               Address of error handling routine (0008)
  4               Channel type ("M" or "M"+128 for 'ad-hoc' channels)
  5               Address of output subroutine
  7               Address of input routine
  9               Length of channel (595)
 11      CHBYTE   Record pointer (0-512).
 13      CHREC    Record number, lower byte
 14      CHNAME   10 byte filename with trailing spaces
 24      CHFLAG   Flag byte: bit  0    set   - open for write
                                       reset - open for read
 25      CHDRIV   Drive number
 26      CHMAP    - Sector number of parent directory on flash disk, or
                  - File handle of a file on the 'server disk'
 28               File type literal.
 29               Record number, upper byte.
 30               Device code

 ..               Not used

 67      RECFLG   Flag byte: bit 0 = 0
                              bit 1 = last record
                              bit 2 = not a PRINT file
 68      RECNUM   Not used
 69      RECLEN   Number of bytes of data in the current record (0-512)
 71      RECNAM   Not used
 81      DESCHK   Not used
 82      CHDATA   512 bytes of data
594      DCHK     Not used
---------------------------------------------------------------------
```

4.3 The 'Work RAM'
is a 512 bytes RAM area, mapped at address #2E00 of the 'Shadow ROM'

4.3.1 Main logical drive descriptor (12 bytes)

#2E00	CRT_DV	Current device code
#2E01	CRT_DR	Current drive number
#2E02	PRV_DV	Previous device code
#2E03	PRV_DR	Previous drive number
#2E04	VOL_OF	Volume offset (in logical drives)
#2E06	CLU_SZ	Cluster size - 1
#2E07	ROOT_D	First sector number of root directory
#2E08	SEC_NR	In-cluster sector number
#2E0A	FAT_SN	Pointer to the current FAT sector number

4.3.2 Alternate logical drive descriptor (12 bytes)
Same structure as the main descriptor

#2E0C	ALT_DV	

4.3.4 Device capacity table (8 bytes)
Size (in logical drives) and number (0-15)
of the last volume for each device

#2E18	VOL_TB	Not used
#2E1A		Server drive
#2E1C		Flash drive
#2E1E		Not used

4.3.5 Volumes table (4 bytes)
Current volume number for each device

#2E20	CRT_VL	Not used
#2E21		Server drive
#2E22		Flash drive
#2E23		Not used
#2E24		Spare initialized variables

4.3.6 Sector buffer pointers (16 bytes)

#2E30	SECT_0	Pointer for buffer 0
#2E32	SECT_1	Pointer for buffer 1
#2E34	SECT_F	Pointer for buffer F
#2E36	SECT_3	Pointer for buffer 3
#2E38	SECT_L	Pointer for buffer L
#2E3A	SECT_S	Pointer for buffer S
#2E3C	SECT_A	Pointer for buffer A
#2E3E	SECT_Z	Parent directory pointer

```
4.3.7   'Server drive' file descriptor (16 bytes)
        #2E40   N_DESC  File type
        #2E4B   N_HNDL  File handle
        #2E4D   N_FLEN  File length

4.3.8   'Input tape' variables (10 bytes)
        #2E50   L_FLAG  Flag
        #2E51   L_FSEC  First sector
        #2E53   L_FPNT  File pointer
        #2E55   L_LENL  Length (low)
        #2E57   L_LENH  Length (high)
        #2E58   L_DEVN  Device code
        #2E59   L_DRVN  Drive number

4.3.9   'Output tape' variables (10 bytes)
        #2E5A   S_FLAG  Flag
        #2E5B   S_FSEC  First sector
        #2E5D   S_DIRN  Directory nr
        #2E5F   S_BLEN  Tape length
        #2E62   S_DEVN  Device code
        #2E63   S_DRVN  Drive number

4.3.10  (.Z80) snapshot variables (10 bytes)
        #2E64   Z_FLAG  Flag
        #2E65   Z_FSEC  First sector
        #2E67   TMP_HL
        #2E69   TMP_JP
        #2E6A   TMP_AD
        #2E6C   Z_DEVN  Device code
        #2E6D   Z_DRVN  Drive number

4.3.11  Printer buffer pointer (2 bytes)
        #2E6E   PBF_PT

4.3.13  Directory search variables (6 bytes)
        #2E70   D_NUMB  Directory number
        #2E72   D_SECT  Sector number
        #2E74   D_PNTR  Pointer

4.3.14  Flash drive block number (4 bytes)
        #2E76   BLK_LO  Low word
        #2E78   BLK_HI  High word
        #2E7A   DAT_LN  Data length

4.3.16  Other
        #2E7C   SPARE_  Spare
        #2E7E   AX_CMD
        #2E7F   AX_ERR

4.3.12  Copy/Rename destination file parameters (16 bytes)
        #2E80   DST_TY  File type
        #2E81   DST_LN  Filename length
        #2E83   DST_AD  Filename address
        #2E85   DST_NA  Filename

4.3.17  Internal Stack (32 levels)

4.3.18  Printer buffer (64 bytes)
        #2EC0   PR_BUF

4.3.19  Page buffer  (256 bytes)
        #2F00   BUFF_P


4.4     Sector buffers (4 KB)

4.4.1   Sector buffers
        #3000   BUFF_0  Main sector
        #3200   BUFF_1  Allocation
        #3400           Main FAT
        #3600   BUFF_3  Work
        #3800   BUFF_L  'Input tape'
        #3A00   BUFF_S  'Output tape'
        #3C00           Alternate FAT

4.4.2   Application data (256 bytes)
        #3E00   AP_DAT
```

4.4.3 Current directories table
Stores the last 64 directory numbers used
#3F00 DIR_TB

5. File system

5.1 Local file system (Flash drive)

- The device is implicitly partitioned into fixed-sized logical drives of 32 MB (65536 sectors of 512 bytes).
- The maximum usable device space is of 128 GB, subdivided in 'volumes' of 255 logical drives.
- The logical drives are formatted according to a simplified 16-bit FAT system, the FAT entries being sector, rather than cluster numbers.
- The allocation unit (cluster) can be of: 2, 4, 8 or 16 sectors
- Sector number 0 of any logical drive is not used.
- The FAT contains (65536 / cluster size) entries, occupying sectors 1 through (256 / cluster size).
- Sector number (256 / cluster size) + 1 contains the first record of the root directory.
- As cluster 0 is always occupied by the FAT, the corresponding FAT entry (bytes 0 and 1 of sector 1) is used to store the cluster size.
- As sectors 0 and 1 are not available for allocation, the corresponding FAT entry values are used for marking:
  0000 = Free cluster
  0001 = Last cluster of the file
- When a file is deleted, the parent directory is compacted by reclaiming the corresponding entry and shifting all further entries to its right, which can span several records, downwards by 16 bytes.

5.2 Remote file system ('Server drive')

The remote device is accessed via 'command blocks' sent to the peripheral controller, which relays them to a machine, running a suitable server application.
The structure of the command block is:

Byte Nr.	Description
1	Command byte: bits 0-3 = command parameter bits 4-7 = command code
2	Auxiliary command code
3,4	Length of data: n = 0-512
5..(n+4)	(n) bytes of data

After processing the command, the server sends back a 'response block':

Byte Nr.	Description
1	Error code or: 0 = No error
2	Auxiliary error code
3,4	Length of data: n = 0-512
5..(n+4)	(n) bytes of 'response' data

5.2.1 Following commands are implemented:

Code	Command	Parameter	Data	Response
0	Close File	Handle		
1	Read sector	Handle		Sector
2	Write sector	Handle	Sector	
3	Set file pointer	Handle	Position	
3	Set pointer to EOF	Handle		File size
4	Create temp. file	Type	Name	Handle
5	Create perm. file	Type	Name	Handle
6	Open temp. file	Type	Name	Descriptor
6	Open next file	any		Descriptor
7	Open perm. file	Type	Name	Descriptor
8	Find file	Type	Name	Descriptor
8	Find next file	any		Descriptor
9	Delete file	Type	Name	
10	Rename file	Type	Name	
11	Copy file	Type	Name	
12	Get directory list	0	Name	List
12	Next directory list	0		List
12	Host command	1	Command	Result
12	Dump OpSys	6	Sector	
12	Download OpSys	7		Sector
12	Print buffer	8	Buffer	
13	Select directory	0	Name	Path

13	Get path	0		Path
14	Create directory	0	Name	
15	Delete directory	0	Name	

5.2.2 Notes
- A 'Handle' is a number in the range: 0-15
- The 'Type' is a number in the range: 0-15 defined at (3.4)
- A 'permanent file' is allocated a handle = 1-15. The same handle is not re-allocated before the file is explicitly closed.
- A 'temporary file' is always allocated the handle = 0. Creating or opening another 'temporary file' automatically closes the previous one.
- For the 'Set file pointer' command the position can be specified either on two, or on four bytes: [<Pos>]<Rec>, where <Pos>= record pointer (0-511) and <Rec>= record number (0-32767).
- The 'Set pointer to EOF' command returns the length of the file as a 3-bytes number.

5.2.3 The 'Find (next) file' command returns a 16-bytes descriptor:

Offset	Length	Description
0	1	File type (0-11)
1	10	File name
11	2	00 00
13	3	File length

5.2.4 The descriptor returned by the 'Open temporary|permanent| next file'command contains also the file handle

Offset	Length	Description
0	1	Actual file type (0-11)
1	10	Actual file name
11	2	File handle (0-15)
13	3	File length

5.2.5 The handle returned by the 'Create temporary|permanent file' command is a two-byte number in the range 0-15.
- After a 'Find file' or 'Open temp. file' command, a subsequent 'Find next file' or 'Open next file' command will attempt to find/open the next file with a matching name.
- A copy|rename operation requires two steps:
  1. The source file is found by issuing a 'Find [next] file' command.
  2. The destination name is specified in a 'Rename file' or 'Copy file' command.

5.2.6 The list returned by the 'Get|Next directory list' command consists of a sequence of 16-bytes file descriptors, with #FF as an end marker. For file types 4-11 the descriptors are the same as those returned by the 'Find (next) file' command. (see 5.2.3)
For file types 0-3 the descriptors contain also information from the 9-byte BASIC header of the files:

Offset	Length	Description
0	1	File type (0-3)
1	10	File name
11	1	File type (from BASIC header)
12	2	File length
14	2	Start line (Program), or Array name (Numeric or String), or Loading address (Code)

5.3 Flash drive
- The flash drive is accessed in the same way as the 'server drive', the command and response blocks having the same structure.
- Following commands are implemented:

Code	Command	Parameter	Data	Response
15	Receive - Write	5	Address	
15	Set write address	6	Address	
15	Read - Transmit	7	Address	
15	Read block	8	Address	Sector

15	Read to buffer	9	Address	
15	Write block	10	Sector	
15	Write from buffer	11	Address	
15	Identify card	12		Identifier
15	First erase address	13	Address	
15	Last erase address	14	Address	
15	Erase block	15		

- 'Address' is a 4-bytes sector number
- 'Identifier' is an 8-bytes card identifier string followed by the 4-bytes capacity (last sector number) of the card
- 'Sector' is a 512-bytes block of data
- Writing a sector requires two steps:
  1. The sector number is specified in a 'Set write address' command
  2. The data is sent via a 'Write sector' command
- The 'Write from buffer' writes the sector read by a previous 'Read to buffer' command
- Erasing a block requires three steps:
  1. The start of the block is specified in a 'First erase address' command
  2. The end of the block is specified in a 'Last erase address' command
  3. The 'Erase block' command is sent to actually erase the block

6. Hook Codes

The ZX INTERFACE 1 hook codes.

Nr	Label	Address	Description	
#1B	WAI_KY	#0103	Console input	
#1C	PRI_NT	#0106	Console output	
#1D		#0109	R232 input	Not implemented
#1E		#010C	R232 output	Not implemented
#1F	L_PRNT	#010F	Printer output	
#20	TST_KY	#0112	Keyboard test	
#21	HK_NOP	#0115	Select drive	
#22	OPN_FL	#0118	Open file	
#23	CLO_FL	#011B	Close file	
#24	ERA_FL	#011E	Delete file	
#25	RD_SQE	#0121	Read sequential	
#26	WR_SQE	#0124	Write sequential	
#27	RD_REC	#0127	Read random	
#28	RD_CSC	#012A	Read sector	
#29	NEXT_S	#012D	Read next	
#2A	WR_CSC	#0130	Write sector	
#2B	CRE_CH	#0133	Create buffer	
#2C	RCL_CH	#0136	Delete buffer	
#2D		#0139	Open network channel	Not implemented
#2E		#013C	Close network channel	Not implemented
#2F		#013F	Get packet	Not implemented
#30		#0142	Send packet	Not implemented
#31	HK_NOP	#0145	Create system variables	
#32	SERV_R	#0148	Call shadow ROM routine	
#33		#014B	Read next header	Not implemented
#34		#014E	Open "B" channel	Not implemented

Additional hook codes

Nr	Label	Address	Description
#35	RD_SEC	#0151	Read sector
#36	WR_SEC	#0154	Write sector
#37	NEXT_R	#0157	Next sector
#38	RCLM_A	#015A	Reclaim 'ad-hoc' channels
#39	LOCT_F	#015D	Find file or directory
#3A	FIND_F	#0160	Get (next) file or directory info
#3B	OPEN_S	#0163	Open stream
#3C	CLOS_S	#0166	Close stream
#3D	SAV_LD	#0169	SAVE / LOAD / VERIFY / MERGE
#3E	MOVE_E	#016C	Copy file or Set file pointer
#3F	CAT_LG	#016F	Catalogue of current or specified path
#40	FORM_T	#0172	Format drive
#41	CLR_SC	#0175	Clear screen
#42	CLOS_A	#0178	Close all streams
#43	SV_CMD	#017B	Custom peripheral port command
#44	A_PATH	#017E	Get absolute path
#45	DIR_FL	#0181	Get first directory list
#46	DIR_NL	#0184	Get next directory list
#47	PR_INT	#0187	Print 3-byte integer
#48	SND_CM	#018A	Peripheral command
#49	RCV_NW	#018D	Peripheral send-receive

- A 'File type literal' is a character as defined at (3.4)
- A 'Device type literal' is a character as defined at (3.1)
- A 'Device code' is a byte defined as:
  - Bits 0-3: Volume number (0-15)
  - Bits 4-6: Device number (0-7)
    - 0 - Not used
    - 1 - Server drive
    - 2 - Flash drive
    - 3 - Not used
  - Bit 7 : 1

6.1 Console input (#1B)

6.1.1 Action: Wait for a key to be pressed

6.1.2 Input data: None

6.1.3 Output data:

- (A) = Character code

```
6.2     Console output (#1C)
6.2.1   Action:  Send a character to the screen
6.2.2   Input data:
        - (A)      = Character code
6.2.3   Output data: None

6.3     Printer output (#1F)
6.3.1   Action:  Print a character to the printer
6.3.2   Input data:
        - (A)      = Character code
6.3.3   Output data: None

6.4     Keyboard test (#20)
6.4.1   Action:  Test if a key is being pressed
6.4.2   Input data: None
6.4.3   Output data:
        - Carry flag set if a key is being pressed

6.5     Open file (#22)
6.5.1   Action: Open a file for sequential access.
6.5.2   Input data:
        - (A)      = #BF (IN)  - Open for read
                   = #DF (OUT) - Open for write
                   = #A5 (RND) - Create a file handle
        - (DRV_NR) = Drive number (1-255)
        - (DEV_TY) = Device type literal or device code
        - (NAM_LN) = Length of filename (1-254)
        - (FIL_TY) = File type literal
        - (NAM_AD) = Address of filename (0-65535)
6.5.3   Output data:
        - (IX)     = Address of the channel descriptor

6.6     Close file (#23)
6.6.1   Action: Close a file.
6.6.2   Input data:
        - (IX)     = Address of the channel descriptor
6.6.3   Output data: None

6.7     Delete file (#24)
6.7.1   Action: Delete a file
6.7.2   Input data:
        - (DRV_NR) = Drive number (1-255)
        - (DEV_TY) = Device type literal or device code
        - (NAM_LN) = Length of filename (1-254)
        - (FIL_TY) = File type literal
        - (NAM_AD) = Address of filename (0-65535)
6.7.3   Output data: None

6.8     Read sequential (#25)
6.8.1   Action: Read the next record
6.8.2   Input data:
        - (IX)     = Address of the channel descriptor
6.8.3   Output data: None

6.9     Write sequential (#26)
6.9.1   Action: Write the current record
6.9.2   Input data:
        - (IX)     = Address of the channel descriptor
6.9.3   Output data: None

6.10    Read record (#27)
6.10.1  Action: Read the current record
6.10.2  Input data:
        - (IX)     = Address of the channel descriptor
6.10.3  Output data: None

6.11    Read sector (#28)
6.11.1  Action: Read sector CHREC into channel buffer
6.11.2  Input data:
        - (IX)     = Address of the channel descriptor
6.11.3  Output data:

6.12    Read next (#29)
6.12.1  Action: Read next sector into channel buffer
6.12.2  Input data:
        - (IX)     = Address of the channel descriptor
6.12.3  Output data:
```

```
6.13    Write sector (#2A)
6.13.1  Action: Write channel buffer to sector CHREC
6.13.2  Input data:
        - (IX)     = Address of the channel descriptor
6.13.3  Output data:

6.14    Create buffer (#2B)
6.14.1  Action: Create a channel descriptor
6.14.2  Input data:
        - (A)      = #BF (IN)  - Open for read
                   = #DF (OUT) - Open for write
                   = #A5 (RND) - Create a file handle
        - (DRV_NR) = Drive number (1-255)
        - (DEV_TY) = Device type literal or device code
        - (NAM_LN) = Length of filename (1-254)
        - (FIL_TY) = File type literal
        - (NAM_AD) = Address of filename (0-65535)
6.14.3  Output data:
        - (IX)     = Address of the channel descriptor

6.15    Delete buffer (#2C)
6.15.1  Action: Delete a channel descriptor
6.15.2  Input data:
        - (IX)     = Address of the channel descriptor
6.15.3  Output data: None

6.16    Create system variables (#31)
6.16.1  Action: Create system variables
6.16.2  Input data: None
6.16.3  Output data: None

6.17    Execute code (#32)
6.17.1  Action: Execute code from address (HD__11)
6.17.2  Input data:
        - (HD__11) = Address of the executable code
6.17.3  Output data: None

6.18    Read sector (#35)
6.18.1  Action: Read sector into buffer.
        - Does not apply to the 'server drive'.
6.18.2  Input data:
        - (A)      = Drive number (1-255)
        - (BC)     = Sector Number (0-65535)
        - (HL)     = Buffer address (0-65536)
        - (DEV_TY) = Device type literal or device code
6.18.3  Output data: None

6.19    Write sector (#36).
6.19.1  Action: Write sector from buffer.
        - Does not apply to the 'server drive'.
6.19.2  Input data:
        - (A)      = Drive number (1-255)
        - (BC)     = Sector Number (0-65535)
        - (HL)     = Buffer address (0-65536)
        - (DEV_TY) = Device type literal or device code
6.19.3  Output data: None

6.20    Next sector (#37).
6.20.1  Action: Find the sector number of the next record of a file.
        - Initially it should be called with (BC)=0 to flush the FAT
          buffer.
        - Does not apply to the 'server drive'.
6.20.2  Input data:
        - (A)      = Drive number (1-255)
        - (BC)     = Sector number (0-65535)
        - (DEV_TY) = Device type literal or device code
6.20.3  Output data:
        - Zero flag set = No more records
        - (BC) = Next sector number (0-65535)

6.21    Reclaim all 'ad-hoc' channels (#38).
6.21.1  Action: All channels not associated with streams are
                reclaimed.
6.21.2  Input data:  None
6.21.3  Output data: None

6.22    Find file or directory (#39).
```

```
6.22.1  Action: Check if a specified file or directory exists.
6.22.2  Input data:
        - (DRV_NR) = Drive number (1-255)
        - (DEV_TY) = Device type literal or device code
        - (NAM_LN) = Length of filename (1-254)
        - (FIL_TY) = File type literal
        - (NAM_AD) = Address of filename (0-65535)
6.22.3  Output data:
        Carry Flag reset = file exists.

6.23    Get (next) file or directory info (#3A).
6.23.1  Action: Retrieve file information.
        - If bit 6 of FLAGS3 is set, the next matching file will be
          found.
6.23.2  Input data:
        - (DRV_NR) = Drive number (1-255)
        - (DEV_TY) = Device type literal or device code
        - (NAM_LN) = Length of filename (1-254)
        - (FIL_TY) = File type literal
        - (NAM_AD) = Address of filename (0-65535)
6.23.3  Output data:
        - Carry Flag reset = file exists.
        - (HD__00) = File type
        - (HD__0B) = Sector number of first record.
        - (HD__0F) = File length (bytes 1 and 2)
        - (HD__11) = File length (byte 3)
        - (DE)     = Directory number
        - (BC)     = Sector number of the directory entry.
        - (HL)     = Pointer to the directory entry.

6.24    Open stream (#3B).
6.24.1  Action: Open a stream to a file.
6.24.2  Input data:
        - (A)      = #BF (IN)  - Open for read
                   = #DF (OUT) - Open for write
                   = #A5 (RND) - Create a file handle
        - (DRV_NR) = Drive number (1-255)
        - (STR_NR) = Stream number (0-15).
        - (DEV_TY) = Device type literal or device code
        - (NAM_LN) = Length of filename (1-254)
        - (FIL_TY) = File type literal
        - (NAM_AD) = Address of filename (0-65535)
6.24.3  Output data: None.

6.25    Close stream (#3C).
6.25.1  Action: Close a stream.
6.25.2  Input Data:
        - (A) = Stream number (0-15)
6.25.3  Output data: None

6.26    SAVE / LOAD (#3D).
6.26.1  Action:
        - Read/Write memory contents from/ to a file.
        - Create (SAVE) / change (LOAD) directory.
        - Close the 'input tape', 'output tape' or 'snapshot file'.
6.26.2  Input data:
        Operation type:
          - (A)    =  0,4 - SAVE
                   =  1,5 - LOAD
                   =  2,6 - VERIFY
                   =  3,7 - MERGE
                   =  8   - Close 'output tape'
                   =  9   - Close 'input tape'
                   = 10   - Close snapshot file.
        File parameters, specified by a descriptor or a handle
          Descriptor:
          - (DRV_NR) = Drive number (1-255)
          - (STR_NR) = 255
          - (DEV_TY) = Device type literal or device code
          - (NAM_LN) = Length of filename (1-254)
          - (FIL_TY) = File type literal
            Effective only for data type: 7,8,10 and 11
          - (NAM_AD) = Address of filename (0-65535)
          Handle:
          - (STR_NR) = Handle (0-15)
        Data type code
        - (HD__00) =  0 - BASIC program
                   =  1 - BASIC number array
```

```
                    =  2 - BASIC string array
                    =  3 - BASIC 'CODE' block
                    =  7 - Binary block
                    = 10 - Tape file
                    = 11 - Snapshot file
          Parameters of BASIC program   only for data type: 0
          - (HD__11) = Auto-run line number
          Parameters of BASIC array,  only for data type: 1-2
          - (HD__0F) = Array name ("a"-"z").
          Length of memory block, only for data type: 3 and 7
          - (HD__0B) = Length of memory block.
          Address of memory block, only for data type: 3 and 7
          - (HD__0D) = Address of memory block.
          File pointer, only for data type: 7
          - (HD__0F) = Record pointer (0-511)
          - (HD__11) = Record number  (0-32767)
            If bit 2 of the 'operation type' in the A-register is set, the
            pointer doesn't need to be specified as it is initialized to 0
            The pointer is automatically updated after the operation
6.26.3    Output data: None

6.27      Copy /rename file (#3E).
6.27.1    Action: Copy or rename files / rename directory
6.27.2    Input data:
          Operation type:
            - (A)    = #CC (TO) - copy
                     = #AC (AT) - rename
          Source: specified by a stream or a file descriptor
            Stream:
            - (STR_NR) = Stream number (0-15)
            File descriptor:
            - (DRV_NR) = Drive number (1-255)
            - (STR_NR) = 255
            - (DEV_TY) = Device type literal or device code
            - (NAM_LN) = Length of filename  (1-254)
            - (FIL_TY) = File type literal
            - (NAM_AD) = Address of filename (0-65535)
          Destination, specified by a stream or a file descriptor
            Stream:
            - (STR_N2) = Stream number (0-15)
            File descriptor:
            - (DRV_N2) = Drive number (1-255)
            - (STR_N2) = 255
            - (DEV_T2) = Device type literal or device code
            - (NAM_L2) = Length of filename  (1-254)
            - (FIL_T2) = File type literal
            - (NAM_A2) = Address of filename (0-65535)
6.27.3    Output data: None

6.28      Set file pointer (#3E).
6.28.1    Action: Set file the pointer of a M or H channel to which
                  a given stream is opened.
6.28.2    Input data:
          - (A)      = Operation type: #A9 (POINT)
          - (STR_NR) = Stream number (0-15)
          - (HD__0F) = Record pointer (0-511)
          - (HD__11) = Record number  (0-32767)
6.28.3    Output data: None

6.29      Catalogue (#3F).
6.29.1    Action: Produce a file catalogue.
6.29.2    Input data:
          - (DRV_NR) = Drive number (1-255)
          - (STR_NR) = Stream number (0-15)
          - (DEV_TY) = Device type literal or device code
          - (NAM_LN) = 0
          A directory or file name may be specified to be used as a
          filter for command's output
          - (NAM_LN) = Length of name (1-254)
          - (FIL_TY) = File type literal
          - (NAM_AD) = Address of name (0-65535)
6.29.3    Output data: None

6.30      Format logical drive (#40).
6.30.1    Action: Format logical drive
          - Does not apply to the 'server drive'
6.30.2    Input data:
          - (A)      = Drive number (1-255)
```

```
        - (STR_NR) = Cluster size: 2, 4, 8 or 16
        - (DEV_TY) = Device type literal or device code
6.30.3  Output data: None

6.31    Clear Screen (#41)
6.31.1  Action: Same as the extended BASIC 'CLS #' Command
6.31.2  Input data: None
6.31.3  Output data: None

6.32    Close all streams (#42)
6.32.1  Action: Same as the extended BASIC 'CLEAR #' command
6.32.2  Input data: None
6.32.3  Output data: None

6.33    Peripheral Module Command  (#43)
6.33.1  Action: Sends a command to the server or the peripheral
                controller and prints the response as hex-dump
6.33.2  Input data:
        - (A)      = Selector
                     bit 0 - Adaptor / not Server
                     bit 1 - Print hex-dump
        - (BC)     = Length of command string
        - (HL)     = Address of command string
6.33.3  Output data: None

6.34    Get absolute path (#44).
6.34.1  Action: Get parameters of absolute path name
6.34.2  Input data:
        - (DRV_NR) = Drive number (1-255)
        - (DEV_TY) = Device type literal or device code
        Relative path name
        - (NAM_LN) = Length of filename (1-254) or
                     Zero for the current directory
        - (NAM_AD) = Address of filename (0-65535)
6.34.3  Output data: Absolute path name in BUFF_3
        - (HL)     = Address of absolute file name
        - (BC)     = Length  of absolute file name

6.35    Get first directory list (#45).
6.35.1  Action: Get the first directory list as
                specified at 3.4.5 and 5.2.6
6.35.2  Input data:
        - (DRV_NR) = Drive number (1-255)
        - (DEV_TY) = Device type literal or device code
        Relative path name
        - (NAM_LN) = Length of directory name (1-254) or
                     Zero for the current directory
        - (NAM_AD) = Address of directory name (0-65535)
6.35.3  Output data: First directory list in BUFF_3
        - (HL)     = Address of first directory list
        - (SER_FL) = 0 if root directory

6.36    Get next directory list (#46).
6.36.1  Action: Get the next directory list as specified
                specified at 3.4.5 and 5.2.6
6.36.2  Input data: None
6.36.3  Output data: Next directory list in BUFF_3
        - (HL)     = Address of next directory list

6.37    Print integer (#47).
6.37.1  Action: Print the 3-byte integer (A) (DE)
                with 3, 6 or 8 digits and leading spaces
6.37.2  Input data:
        - (DE)    = Lower bytes
        - (A)     = Upper byte
        - The flags specify the number of digits:
          ---------------------
            Zero  Carry  Width
          ---------------------
           reset    -      3
            set    set     6
            set   reset    8
          ---------------------
6.37.3  Output data: None

6.37    Peripheral command (#48)
6.37.1  Action: Send only a command code to the peripheral
                controller.
```

6.37.2 Input data:
- (A) = Command code
6.37.3 Output data:
- (A) = Error code (0 = no error)
- (HL) = Adress of response data (0-65535)
- Carry = Set if error

6.38 Peripheral send-receive (#49)
6.38.1 Action: Send and receive data to/from the peripheral controller.
6.38.2 Input data:
- (A) = Command code
- (HL) = Data block address (0-65535)
- (BC) = Length of data block (0-512)
- (DE) = Response address (0-65535)
6.38.3 Output data:
- (A) = Error code
- (HL) = Response address
- (BC) = Response length