WHAT IS

EXMATRIX

USER MANUAL

ZXMATRIX Interface

USER MANUAL

By Sami Vehmaa 2008-2010 rev 1.4

Contents

Introduction:

Thanks for supporting my work on interfaces, I hope this interface will bring you a lot of fun for many years to come.

This is something new for the ZX Spectrum range of computers, it adds USB (universal serial bus) HOST interface to your computer.

What does it offer ? It depends on the configuration you bought it in.
(By having same pcb for 2 different interfaces I can still offer a lowcost CF interface)

1. **USB & CF**	Choose by jumper in what mode you wan't to run it in. (Possible in future both USB and CF socket works at same time.)
2. **USB**	
3. **CF**	(Do note that this is one slot only, you need +2 upg. module for second slot with optional Kempston Joystick interface.)

The interface is a ***autoconfig***, no jumper to be set for spectrum model.
1024KB onboard memory.

On USB there are 2 USB type A connectors, slot 1 & 2, at present only slot 2 will work with one drive connected, this is a software limit and hopefully will change in the future.

Please visit my ZX Spectrum DIY page at regular basis for news/information.

ZXMATRIX Interface is a platform for both USB & CF, on CF mode ResiDOS versions ZXCF,ZXCF+2 can be used. In USB mode ResiDOS ZXUSB is used.

REMEMBER TO CHECKOUT MY SOFTWARE PACKAGE FOR RESIDOS

USBstick & CFcard is refered as flashdrive, ZXCF,ZXCFk,ZXCF+,ZXCF+2 are refered as ZXCF

This manual has been updated to ResiDOS v2.25

Installing for the first time

Installing *ResiDOS* is simple. First download the appropriate version of the installation software for your interface and load it into your Spectrum. The software is provided in the form of an emulator TAP or TZX file, so you will need to transfer it using software such as *playtzx* or an emulator. You'll find suitable software on the World Of Spectrum site. Once it's transferred, just LOAD it into your Spectrum and follow the simple on-screen instructions. (on page 43 there is some info on MP3 versions)

During installation, the program will confirm the size of your interface's RAM.

Once you have changed the jumpers/switch on your interface as instructed, the Spectrum will reset and you should see the *ResiDOS* boot screen appear!

Upgrading to a new version of ResiDOS

When a new version of *ResiDOS* becomes available, installation is even easier. Just load the installation program from tape as normal, and it will automatically detect your existing version. You can then choose to upgrade, and the system will be automatically updated, leaving all your installed ROMs and tasks still in the interface's memory.

If you choose to re-install, the normal installation procedure will occur. This will wipe all the packages, ROMs and tasks from your interface's memory.

If you have the FATfs package installed, then instead of loading from tape you can simply copy the installation files onto your FAT16 flashdrive, and LOAD them from there.

Setting up your flashdrive

From v2.00, *ResiDOS* can work with flashdrives formatted to one of two different filesystems:

- IDEDOS, including +3DOS
- FAT

You can use any combination of IDEDOS and FAT flashdrives. If you have two flashdrives, you can format one with IDEDOS and one with FAT. If you have only one flashdrive, you can choose either system. It is also possible to divide a single flashdrive between IDEDOS and FAT.

IDEDOS or FAT?

FAT has the following advantages:

- All PCs and Macs can read and write to FAT flashdrive, so transferring files between your PC and Spectrum is extremely easy
- Each partition can be up to 2GB in size (when FAT32 support is added, even this limitation will be removed); IDEDOS partitions are limited to 16MB (although there is no limit on the number of partitions)
- Files can be organized into directories

IDEDOS has the following advantages:

- IDEDOS drives can be directly used by the +3e
- IDEDOS drives can be used with the ZXVGS operating system
- CP/M, which may be ported to *ResiDOS* soon, needs IDEDOS drives
- Swap partitions, which could be used by games and other apps, are only currently available on IDEDOS drives

Setting up a FAT flashdrive

ResiDOS cannot directly format FAT flashdrives. Instead, format your flashdrive with your PC. You need one or more FAT16 partitions. In Windows XP, this is done by selecting "FAT" as the format type; do not select "FAT32", as the *FATfs* package does not support it at the time of writing.

Then simply insert the flashdrive into your interface and reset the Spectrum. It should be automatically detected and mapped to one or more drive letters (shown on the startup screen).

Setting up an IDEDOS flashdrive

Before you can use your flashdrive, it needs to be initialised with the IDEDOS system. ResiDOS can work with two different physical flashdrives, normally known as the "master" and "slave" unit; in all the following commands, the first number given identifies the unit number, which is 0 for the master and 1 for the slave.

The first thing to do is make sure that *ResiDOS* understands the structure of your flashdrive. Normally, it automatically detects this and displays the details on the boot screen. However, if for some reason this doesn't work, you can manually specify the parameters (shown on the flashdrive itself, or to be found on the manufacturer's website) with the following command:

- **%DRIVE 0,977,10,17**

(The above command assumes your flashdrive is the master drive (unit 0) and has 977 cylinders, 10 heads and 17 sectors). Once you have entered this command, the details are permanently stored in your interface's RAM.

You can re-enable auto-detection for the master USB drive (unit 0) with the command:

- **%DRIVE 0**

If you want to disable detection of a flashdrive altogether, to save a bit of time on boot, you can do so with the following example command, which disables detection of the slave flashdrive (unit 1):

- **%DRIVE 1,0,0,0**

When you're sure that *ResiDOS* knows the correct details, you need to initialise the flashdrive. **Note that doing this erases everything currently stored on the flashdrive**. The command to use is:

- **%REFORMAT 0,15**

This initialises the master flashdrive (unit 0), allowing up to 15 partitions to be created. You can specify any number you like for the number of partitions, but note that the more you allow, the slower certain partition management commands will be. Each partition can be up to 16Mb in size; I'd recommend calculating the number you would need to completely fill the flashdrive, and adding several more for a bit of elbow-room. So, a 1Gb flashdrive could use about 1000/16=63 partitions, and you might wish to specify around 80.

From v1.92, it is possible to share a flashdrive between *ResiDOS* and another filesystem (such as FAT). To do this, you need to limit *ResiDOS* to using a limited number of cylinders on the flashdrive. The command to use is:

- **%REFORMAT 0,15,200**

This formats the master flashdrive as before, but only the first 200 cylinders will be used by *ResiDOS*, leaving the rest available for other filesystems. For more information on how this works, and how to create an effective shared flashdrive, see the explanation on the +3e website about sharing disks (note, however, that the syntax of +3e commands differs from the *ResiDOS* syntax described here).

Once the flashdrive is initialised (you only ever need to do this once) you can use the partition management commands to add some partitions for storing your files.

Switching flashdrive

To switch to using a flashdrive, you can use the following command:

- **%DRIVES**

This will unmap all existing flashdrives and re-detect your flashdrive, automatically mapping any drives back again as it would do on boot. Your BASIC program and all other data will be unaffected by this process, so it's a very useful command for transferring files from one flashdrive to another.

Note that once your flashdrive is initialised, *ResiDOS* stores its details (about cylinders, heads, sectors etc) on the flashdrive itself, so when swapping flashdrive there is no need to set these details again.

Entering ResiDOS Commands

ResiDOS commands are easy to enter under any version of BASIC, from the standard keyword-driven 48K BASIC to the more recent versions such as *SE BASIC* and *Gosh Wonderful*, which allow commands to be entered letter-by-letter.

For *all* commands, the **%** characters shown on this website are completely optional. They're only required if the command would otherwise be accepted by BASIC (eg, in most BASICs, **LOAD "name"** would load a program from tape, so you need to use **LOAD %"name"** to load a program from flashdrive). For BASICs which use keyword entry (eg standard 48K BASIC), it's also usually convenient to start commands which don't have a keyword with **%**, simply to take the Spectrum out of **K** mode and into **L** mode. On single-character-entry BASICs like *SE BASIC* or *Gosh Wonderful* there's no need to bother with the leading **%**.

Additionally, commands which are not present as Spectrum keywords (such as **LOAD** and **SAVE**) may be entered as upper or lower-case (or a mixture), although they are always shown in capitals on this manual. If you are using a single-character-entry BASIC, then they will usually accept keywords in lower-case as well.

Finally, commands can be abbreviated, to save you from having to type in the entire command name. This is done by adding a "**.**" character to the partial command name. *ResiDOS* will make a guess at the command desired and fill it in for you.

As an example of the flexibility of the system, here are a number of ways to enter the same command (which lists the partitions on your flashdrive):

- **%PARTITIONS**
- **PARTITIONS**
- **partitions**
- **%Partitions**
- **%PAR.**
- **p.**

Entering functions

From v1.81, *ResiDOS* contains its own built-in functions, that can be used as parts of expressions in the same way as other functions. They are entered as if they were user-defined functions, as the **FN** keyword followed by the function name and a list of parameters in brackets. Optionally, you can use a **%** sign between **FN** and the function name. A couple of examples of using *ResiDOS* functions in expressions are:

- **GOTO FN ERL()+1**
- **PRINT 100*FN %ERR()**

Partition Management

Note that some of this section is concerned with creating and managing partitions on IDEDOS flashdrives, so if you are using FAT flashdrives (with the optional FATfs package), then much of it does not apply. However, it is still possible to list partitions on FAT flashdrives with the **%PARTITIONS** command.

Partition management

The IDEDOS system used by *ResiDOS* allows you to divide your flashdrive up into many separate *partitions*, which can each be assigned to a drive letter and used for storing files. You can have as many partitions as you like on your flashdrive, but due to memory constraints it may not be possible to map them all at one time.

Partitions can be one of two types: "data", for holding files; and "swap", for use as temporary workspace by programs or the operating system. Other types may be added in the future. Note that no programs currently require swap partitions, but there is no harm in creating one for future use.

Commands exist to allow you to create and delete partitions and assign them to drive letters. Unlike some inferior operating systems, this can all be done from BASIC, and there is no need to reset your Speccy at any point!

Creating, deleting and listing partitions

You can create a partition using one of the following commands, which create data and swap partitions respectively:

- **%PARTITION "***name***",***size*
- **%PARTITION "***name***",***size***, 2**

To identify which unit a partition is to be created on, the number can be included at the start of the name, followed by the unit number (eg "0>Documents" or "1>Games"). If you don't specify the unit, then unit 0 is assumed. Names can be up to 16 characters long (this doesn't include the unit identifier) and are not case-sensitive.

The size for a partition can be given in kilobytes (K) or megabytes (MB), with a maximum size of 16 megabytes allowed. A number smaller than 256 is assumed to be a size in megabytes; a number larger or equal to 256 is assumed to be a size in kilobytes.

You can delete a partition, together with any files that are stored on it, using the command:

- **%ERASE "***unit>name***"**

In this command, you *must* specify the unit number, as otherwise it will be assumed you are trying to erase a file. You can use **ERA** or **DEL** instead of **ERASE** if you prefer. Note that a partition can only be deleted if there are currently no drives mapped to it.

You may also rename a partition, using the following command:

- **%REN "***unit>name***","***newname***"**

Again, you *must* include the unit identifier for this command, otherwise it will assume you want to rename a file, not a partition.

Finally, you may display a list of partitions, in a normal or expanded form, using the following commands:

- **%PARTITIONS**
- **%PARTITIONS+**

This will show all partitions on all units. Note that if you have other filesystem packages installed (such as *FATfs*) then some drive will show up more than once, with different unit numbers.

Automatic drive mappings

Normally, when the Spectrum starts up, *ResiDOS* automatically maps all the data partitions it finds to all the available drive letters. If you don't like this behaviour, you can manually specify your own drive mappings. You can also turn off the automatic mapping feature with the command:

- **%AUTOMAP-**

The feature can be re-enabled at a later time with the command:

- **%AUTOMAP+**

Mapping drives to partitions

It is also possible to manually map drives to partitions, so that you can choose which drive letters a particular partition uses.

When mapping a drive to a partition, you can make the assignment permanent, which means that *ResiDOS* will automatically make the same assignment every time the Spectrum starts up. This is done by adding the + character to the end of the mapping command. Such permanent mappings happen before the automatic mapping takes place, so you can permanently map your favourite drives and let automatic mapping take care of the remainder.

The commands to map drives to partitions are:

- **%MAP "***drive***:","***name***"**
- **%MAP+ "***drive***:","***name***"**

Similar commands are available to remove mappings from drives. Using the + character here means that any permanent assignment associated with the drive will also be removed:

- **%UNMAP "***drive***:"**
- **%UNMAP+ "***drive***:"**

Finally, you can obtain a listing of the current drive mappings with the following command, in which you can optionally specify a stream number for the output (ie stream 3 is often used for the printer):

- **%MAP**
- **%MAP #3**

Drives, directories, paths and user areas

The location of a file is determined by one or more of the following pieces of information:

- drive (a letter from A-P)
- user area (a number from 0-15)
- path (a list of directories, separated with the / or \ character)

Drives are available for all filesystems. User areas are only supported by IDEDOS drives. Directories and paths are only supported by FAT drives.

You can specify any, all or none of the pieces of information above when naming a file. Any missing information is provided by the current default drive, user area or path, which can be changed using the **%CD** command (see later). When specifying locations, the user area must come first, followed by the drive, followed by a colon (:), followed by the path. For example, the following are all valid locations:

- A:
- 2:
- \
- 0e:
- ../
- docs/letters/
- f:\zcode\infocom\zork

Locations can be included with filenames in any command that uses them (eg SAVE, LOAD, %CP; see files for more information).

Setting the default drive, user area or path

You can set the drive which is used by default to any you like; if you don't use this facility, then *ResiDOS* will use **A:** as the default drive, and **0** as the default user area. To set the default drive, user area or path, use the commands:

- **%CD "c:"**
- **%CD "3:"**
- **%CD "5d:"**
- **%CD "games/"**
- **%CD "e:\data\new"**

If you wish, you can make your preference permanent by adding the + character to the end of the command. For example, to make E: the default drive, and have *ResiDOS* remember this preference every time you switch on, use:

- **%CD+ "E:"**

To make a path permanent, you *must* specify the drive to which it applies. Each drive has its own default path, so you can set permanent paths for multiple different drives. For example:

- **%CD+ "A:games/"**
- **%CD+ "e:\data\new"**

Creating and deleting directories

You can create directories with the **%MD** (or **%MKDIR**) command:

- **%MD "advents"**
- **%MKDIR "c:\advents\level9"**

You can delete directories with the **%RD** (or **%RMDIR**) command. They must be empty before they can be removed:

- **%RD "games"**
- **%RMDIR "/data/test"**

Showing the current location

One final command that is available is **%PWD**, which simply prints the current working directory (ie the current user area, drive and path):

- **%PWD**

Using files in ResiDOS

Filenames consist of an 8-character name, optionally followed by a full stop (**.**) and an extension of up to 3 characters. The characters allowed are:

- letters **A** to **Z** (upper- or lower-case; there is no distinction)
- numbers **0** to **9**
- the following characters: **" # $ ' @** ^ **_ { } ~** `

Although extensions are not needed by BASIC or *ResiDOS*, you may find it helpful to name files in a consistent way, so that all BASIC programs have an extension of **.BAS**, for example.

Some example filenames are:

- prog
- TEST.BAS
- puzzle
- screen.scr
- test21.z__

Wildcards

Some commands can have wildcards provided in their filenames. These are used to match multiple files. The following wildcard characters are allowed:

- ***** matches any sequence of characters
- **?** matches any single character

Wildcard characters can be used in either the name or extension part of the filename. Some examples are:

- *.*
- *.bas
- c:test.b??

LOADing and SAVEing files

You can load and save files to and from the flashdrive using the standard **LOAD** and **SAVE** syntax, but adding a **%** sign after the command name. For example:

- **SAVE %"hello.bas" LINE 10**
- **LOAD %"piccy" SCREEN$**

Note that *DATA* files are not currently supported, although BASIC programs, *CODE* files and *SCREEN$* are.

If you attempt to **SAVE** a file when one already exists with the same name, the existing file will be renamed by changing its extension to **.BAK**, before the new file is saved. If there was an existing **.BAK** file of the same name, it will be automatically deleted.

You can also LOAD any type of file that was not created by the Spectrum (standard PC text files, for example), using **LOAD %"name" CODE**. If you do not specify a start address, then 32768 will be used as the default. Such files can alternatively be LOADed using the **SCREEN$** keyword, although this obviously only really makes sense if they are Spectrum screen dumps.

Additionally, snapshot files in the standard .SNA or .Z80 file formats (including those saved by the Task Manager, which are in .Z80 format) can be reloaded using the command:

- **%SNAPLOAD "snapname"**

This command can load 48K and 128K snapshots, although you will get an "out of memory" error if you try to load a 128K snapshot when in 48K mode. Note also that the command decides which format the snapshot is in from the file extension, so be sure to name your snapshots with the appropriate extension (.Z80 or .SNA).

To aid compatiblity with certain snapshots, there are two additional options with this command. Use "+" to make the snapshot loader use a small amount of screen memory (normally it uses the stack area, which is okay in most cases). Use "-" to run the snapshot using the Spectrum's built-in ROM rather than the patched one used with *ResiDOS*. Either or both of these options may be used, eg:

- **%SNAPLOAD+ "snapname"**
- **%SNAPLOAD- "snapname"**
- **%SNAPLOAD+- "snapname"**

Changing BASIC auto-run status

It is not possible to **MERGE** files in *ResiDOS*, but you can change whether or not a BASIC program auto-runs, using the **LINE** command:

- **LINE %"filename",line**

If you specify a line number of 0, then the BASIC program will no longer auto-run.

Cataloguing files

To show a list of all files on the current drive in the current user area, use one of the following:

- **%CAT**
- **%DIR**

You can also specify a string containing a drive, path and/or user area, to see files in that particular location (otherwise the defaults are assumed). The string can also contain a filename or wildcards, and will display only those files which match the string.

Additionally, you can add the + character at the end of the command to show all files (including system files,

which are normally hidden) together with the attributes for each file. Some additional information is also given for BASIC and CODE files saved by *ResiDOS*.

Finally, you can specify a stream number, so that you can (for example) print the catalog listing. (This also applies to all other commands in *ResiDOS* which just produce a listing of information).

Some examples are:

- **%CAT "d:games\"**
- **%DIR "15C:"**
- **%CAT "*.bas"**
- **%CAT+ "e:test.*"**
- **%DIR+ #3;"c:"**

Deleting files

To erase files from the drive, use one of the following commands:

- **%ERASE "filename"**
- **%ERA "filename"**
- **%DEL "filename"**

You can use wildcards in order to erase many files at the same time, but be careful if using this facility!

Copying files

You can make a copy of file to a new file with a different name, or copy one or more files (using wildcards) to a different drive, user area or directory. If you specify "" as the destination, then the current location is used. Some examples are:

- **%CP "letter.doc","letter2.doc"**
- **%CP "*.rom","b:"**
- **%CP "0c:*.*","3c:"**
- **%CP "manic.sna","e:"**
- **%CP "/snaps/sinclair/c/*.z80",""**

Renaming files

To rename a file, use:

- **%REN "filename","newname"**

This must be a single file (you cannot use wildcards). It is possible to use this command to change the user area for a file (but not the drive or directory), simply by specifying the same name with a different user area.

Changing file attributes

You can add or remove various file attributes from a file (or group of files, using wildcards). The possible attributes are:

- **P** (protected, or read-only)
- **S** (system, or hidden)

- **A** (archive - doesn't have any real use)

Protected files cannot be erased or changed, and system files cannot be copied and are not normally listed in catalogs (only in expanded catalogs). You can add or remove a single attribute at a time, by using a "+" or "-" character, as in the following examples:

- **%ATTRIB "*.*","+p"**
- **%ATTRIB "*.sys","+s"**
- **%ATTRIB "letter.doc","-p"**

128K RAMdisk emulation

128K Spectrum BASIC includes a number of commands for saving files to the extra memory as a RAMdisk. *ResiDOS* includes emulation of these RAMdisk commands, which work in both 128K and 48K modes.

Setting up the emulated RAMdisk

ResiDOS always uses user area 15 of drive M: to hold RAMdisk files (it uses the flashdrive rather than memory, and so can store a lot more files than the 62K or so that the 128K Spectrum can manage). In order to use the RAMdisk commands, therefore, you must first map drive M: to a suitable partition. Here's one suggested way to ensure you always have a 500K RAMdisk available whenever you start the Speccy:

- **%PARTITION "RAMdisk",500**
- **%MAP+ "M:","RAMdisk"**

Alternatively, if you want to share a partition that you already use (this is quite safe, as the RAMdisk always uses user area 15, whereas the normal default user area is 0), remap it to drive M: as follows. If this is the drive you want to use by default for normal file operations (on user area 0, for example), you can also do that as shown.

- **%MAP+ "M:","name"**
- **%CD+ "0M:"**

If you are map drive **M:** to a partition on a FAT flashdrive (using the optional FATfs package), then you should be slightly more careful. Since FAT filesystems don't support user areas, using the RAMdisk will affect all files in the current directory of the flashdrive; there is no user area protection, as with *IDEDOS* partitions. So be careful if you decide to do something drastic like **ERASE !"*.*"**.

Using the emulated RAMdisk

Once it has been set up, you can use all the standard 128K BASIC commands that access the RAMdisk. The only limitations are that you cannot currently load or save *DATA* files, or perform **MERGE** operations, as these facilities are not yet available in *ResiDOS*.

Some example commands are:

- **CAT !**
- **LOAD !"name"**
- **SAVE !"name" LINE 10**
- **LOAD !"name" SCREEN$**

Customising ResiDOS

As well as being able to configure *ResiDOS* to automatically set up drive mappings and default drives when you boot up, you can customise it in other ways.

All "permanent" settings are held in a special system file called *residos.cfg* which is saved on the first available partition of your system. This is a text-based configuration file, so if you wish you can edit it on your PC/Mac, as well as using the normal *ResiDOS* commands.

If you switch flashdrive, then a new config file will be created on the new flashdrive. This allows you to keep different settings for different flashdrive (for example, the AUTORUN file will probably need to be different). If you prefer to keep the same settings, you can simply copy the *residos.cfg* file across from one flashdrive to the other.

You can stop the permanent settings (and automapping) from happening if you hold down the CAPS SHIFT key whilst booting the Spectrum. You can also delete all permanent settings in one go by deleting the *residos.cfg* file.

Custom colours

You can choose custom colours which will be set every time the Spectrum starts up, rather than the usual boring old black on white. To do this, use the normal colour commands, but followed by "%". By using the percentage sign you specify that the current colour scheme (including the new colour you are just setting) will be used whenever you boot. Additionally, you can use the new **ATTR** command to specify all colours together (this is calculated as *ink+(paper*8)+(bright*64)+(flash*128)*). The commands are:

- **INK % n**
- **PAPER % n**
- **BRIGHT % n**
- **FLASH % n**
- **ATTR % n**

Note that because the % sets *all* the current scheme as default, the following examples have the same effect, and set a default colour scheme of yellow ink on blue paper:

- **INK % 6:PAPER % 1**
- **INK 6:PAPER % 1**

Autorun files

You can also set an "autorun" file. This must be a BASIC program (which can do whatever you like) which will run automatically on boot, or just whenever you enter the special **%AUTORUN** command.

To set an autorun file, use one of the following commands:

- **%AUTORUN "filename"**
- **%AUTORUN+ "filename"**

You may specify a drive and/or user area with the filename (otherwise the defaults will be assumed). These two commands differ, because with the second case, the autorun file will automatically be loaded every time you reset (or NEW) your Spectrum. The first command stores the filename, but will not load it unless you enter the following command:

- **%AUTORUN**

Note that you can hold the CAPS SHIFT key down when booting to stop drives from being automatically mapped, so any autorun file will not be able to be loaded.

You can disable the autorun file with the following command:

- **%AUTORUN-**

Using optional packages, alternative BASICs and Interface 2 ROMs

The RAM in your interface can be used to store alternative versions of BASIC, or Interface 2 ROM images. Once they have been added, you can start them up at any time using a simple command.

Additionally, *ResiDOS* can be enhanced by loading optional *packages* which provide additional features such as new commands and access to other types of filesystems, such as the common FAT-16 type used on PCs.

Installing ROMs

You can install ROMs in one of two ways:

- either, directly from a file stored on your flashdrive
- or, by first loading the ROM into memory (from tape or flashdrive)

Suitable ROMs for installation include ResiDOS- specific packages, alternative BASICs (such as *Gosh Wonderful*, *SE BASIC* or *Sea Change*) or Interface 2 ROMs (such as those originally released by Sinclair, or unreleased prototypes from Parker which have recently come to light).

Installing ROMs from a file

If you have copied a ROM to flashdrive, then install it with the following commands:

- **CLEAR 32767**
- **%INSTALL "***filename***",***n*

The first command reserves space for the installation process, and the second performs the installation. The name of the ROM kept by ResiDOS will be the filename (excluding any extension after a "."). The number, *n*, should be any of the following values, added together as appropriate:

- 0: for a package, or an Interface 2 ROM (or any other ROM that does not require *ResiDOS* support)
- 1: for a BASIC ROM
- 16: to install the ROM into the writeable "RAM" part of the interface memory (the non-writeable "ROM" part is used by default)
- 32: to disable the NMI button for this ROM (only affects BASIC ROMs)

If you don't specify a value, then the default of 0 will be used. If you are installing a package, then any value except 0 will be ignored, since packages include their own information telling *ResiDOS* how they should be installed.

Note that it is generally advisable to only store BASIC ROMs into the non-writeable "ROM" part of the memory, since unless they have been specifically patched they tend to overwrite themselves.

A couple of examples are:

- **CLEAR 32767**
- **%INSTALL "SEBASIC.ROM",1**
- **%INSTALL "c:jetpac.rom",0**
- **%INSTALL "FATfs.pkg"**

Installing ROMs already loaded into memory

An alternative installation method is to load the ROM image into memory, and then install it directly from there. Since this allows you to load from tape, it is handy for installing packages like *FATfs* when you don't have access to the flashdrive.

To use this method, first CLEAR to a suitable address, and then load in the ROM to above this address. For example:

- **CLEAR 39999**
- **LOAD "FATfs" CODE 40000**

Then the ROM can be installed by using the following form of the **%INSTALL** command:

- **%INSTALL "***name***"@***address***,***n*

As before, the *"n"* value is optional.

To install the package just loaded in the previous example, use:

- **%INSTALL "FATfs"@40000**

Managing alternative ROMs

You can see a list of all the installed ROMs in your interface by using the following commands (the second version includes hidden ROMs, used by the system but not selectable by the user):

- **%ROMS**
- **%ROMS+**

You can uninstall any ROM using the following command (this only removes it from the interface memory, not from the flashdrive):

- **%UNINSTALL "***name***"**

Using alternative ROMs

Once you have installed some ROMs as described in the previous sections, they are available for instant use whenever you switch on the Spectrum. To use the features of an installed package, see the documentation for that package. To switch to an alternative BASIC, or run an Interface 2 ROM, use the following command:

- **%ROM "***name***"**

For example, having installed the ROMs above, we could use either of them like this:

- **%ROM "SEBASIC"**
- **%ROM "jetpac"**

If you specify an Interface 2 ROM, it will immediately start. Pressing the reset button will reset the ROM. To return to *ResiDOS* you will need to either reset your interface and the Spectrum together, or switch off the power completely and then switch the system back on.

If you are activating a BASIC ROM, the Spectrum will restart using that BASIC. You can then use *ResiDOS* commands as normal, and the additional features of the particular BASIC you have chosen.

The new BASIC will remain as the default one even if you switch off the Spectrum and turn back on again. To return to the original BASIC, you can simply select the "Standard" ROM , for example:

- **%ROM "Standard"**

Alternatively, hold down the CAPS SHIFT key while the Speccy is booting up, and the standard ROM will be automatically re-selected.

One final feature is the ability to switch off the interface memory altogether and use your Spectrum's built-in ROM, without having to unplug the interface or set the upload or disable jumper (this is useful for 128K Spectrums). To do this, use the command:

- **%ZX**

Spectrum Modes with ResiDOS

ResiDOS can operate in a number of different modes, depending upon the configuration of your Spectrum. The two basic modes of operation are:

- 128K mode
- 48K mode

When *ResiDOS* starts up, it displays the current mode on the title screen. If you have a 48K Spectrum, then 48K mode is the only possibility. For 128K Spectrums, *ResiDOS* will usually start up in 128K mode.

128K Mode

In *ResiDOS*, 128K mode behaves in the same way as what is often referred to as **USR 0** mode; this mode is usually required by demos, and modern programs from Russia and other countries. (Normally, 128K Spectrums enter this mode by typing **USR 0** in 128K BASIC).

This mode gives access to all the special hardware of the 128K machines (the sound and extra memory), but without access to the 128K BASIC editor and commands. Under *ResiDOS*, the 128K RAMdisk commands and **SPECTRUM** command are also available (in fact, they also work in 48K mode).

This mode is recommended for loading 128K programs. Some 128K programs will not load in this mode (since they check to see if 128K BASIC is active). In such cases, you must disable *ResiDOS* and restart into 128K BASIC with the %ZX command.

IMPORTANT NOTE: The current version of the Task Manager does not understand that some tasks might run in 128K mode and use the whole 128K of memory, and will not save 128K snapshots. It is possible to have a single task loaded which uses the extra memory of the 128K Spectrum, and switch between that and other tasks. However, if two 128K programs are loaded, there will be problems. This will be addressed in a future version.

48K mode

Some games will not work correctly in 128K mode. For these you can use the **SPECTRUM** command to force the computer into 48K mode until the next reset.

If you always want to work in this mode, the command **SPECTRUM+** will make 48K mode active every time the computer is switched on. To disable this mode, and re-enter 128K mode on the next reset, use the **SPECTRUM-** command.

Disabling *ResiDOS*

It is possible to temporarily disable *ResiDOS* and then re-enable it, whilst leaving any program in memory unchanged. This can be useful if you have some piece of hardware which is incompatible with *ResiDOS*, such as Microdrives, or a Wafadrive etc. It is also useful for switching from the **%ZX** mode's 128K BASIC back into *ResiDOS* without having to switch off.

ZXCF range, ZXMatrix in CF mode	*Disable ResiDOS:* **OUT 4287,128**	*Re-enable ResiDOS:* **OUT 4287,64 &**
ZXUSB,	*Disable ResiDOS:* **OUT 63,128**	*Re-enable ResiDOS:* **OUT 63,64**

BREAK and error-trapping

ResiDOS contains facilities to detect BREAK and error conditions, and have them handled by your own BASIC program, instead of simply having an error message produced. This allows you to write more professional-looking BASIC programs, which can't be broken in to, and which can cope with expected errors (such as being unable to find a file requested by the user, for example).

You are advised to **SAVE** your work before running any program with error-trapping facilities, as if you have made a mistake somewhere, much trouble can ensue!

BREAK-trapping

You can prevent the user from BREAKing into your program with the **%ONBREAK** command. This has two forms; to stop BREAK from being effective, use:

- **%ONBREAK THEN CONTINUE**

The **THEN** keyword is optional, and used to make entering the **CONTINUE** keyword easier on normal keyword-driven BASICs. On single-letter-entry BASICs, there is no need to use it.

To restore the normal operation of BREAK, use:

- **%ONBREAK OFF**

Note that BREAK-trapping prevents the user from stopping the program in any of the following ways:

- pressing CAPS SHIFT + BREAK

- pressing SPACE or "n" at a "scroll?" prompt
- entering **STOP** in an **INPUT** statement

Error-trapping

All other errors (except for "OK", "Program finished" and "STOP statement") can be trapped using the **%ONERR** command. This takes any of the following forms, allowing you to transfer control to an error routine, or just call a subroutine before continuing with the program after the point where the error occurred:

- **%ONERR THEN GOTO** *line*
- **%ONERR THEN GOSUB** *line*
- **%ONERR OFF**

(Again, the **THEN** statement is optional.)

Your error-handling routine can use any of the following new functions to determine what the error was. They provide the error number, line and statement respectively:

- **FN ERR()**
- **FN ERL()**
- **FN ERS()**

Additionally, the following command makes the last trapped error occur as normal (as if it had not been trapped):

- **%REPORT**

As an example, here is a program that turns on error trapping, causes an error and lets the error routine output details of the error:

```
  10 %ONERR GOTO 9000
  20 PRINT "Hello"
  30 PRINT: PRINT f
  40 PRINT "Never get here"
8999 STOP
9000 PRINT "Error code was ";FN ERR()
9010 PRINT "Error line was ";FN ERL()
9020 PRINT "Error statement was ";FN ERS()
9030 PRINT "Now causing error..."
9040 %REPORT
```

Extended Channels

ResiDOS provides built-in support for accessing new extended channels in addition to the three standard ("K", "S" and "P") channels. Streams can be opened and closed using the normal **OPEN #** and **CLOSE #** commands (and *ResiDOS* also fixes the bug with the Spectrum's **CLOSE #** command so that it is safe to use). Machine-code programmers also have comprehensive access to these extended channels (see the programming section for details).

Once a stream is open to a channel, you can send output to it and receive input from it using any standard Spectrum BASIC command that allows a stream number to be specified. Additionally, most *ResiDOS* commands that display output may be redirected to a stream, allowing you to capture and manipulate such information. For example, any of these commands are allowed:

```
PRINT #5;"Some text"
INPUT #7;a$
LIST #4;9000
LPRINT #4;x$
LLIST #10

CAT #6;"*.bas"
%roms+ #15
%partitions #8
%pwd #2
%map #4
%tapelist #9
```

Additionally, *ResiDOS* provides two additional functions and a command to read/write the current position in the stream, and to read the size of the stream (not all channels support these operations):

FN ptr#(stream)

FN ext#(stream)

POINT #stream, position

ResiDOS does not actually provide any extended channel types itself; these may be provided by various optional packages. The most important of these is the channels package, which you are encouraged to download and install.

Commands and functions for streams and channels

ResiDOS uses the following commands for manipulating streams and channels:

OPEN #*stream*, *c$*

This command opens a stream to a channel. The value of *stream* can be any number from 0 to 15; if one of the standard streams (0-3) is specified, then its original assignment is replaced with the new channel specified. The value of *c$* is either a single character representing one of the Spectrum's standard channels ("K", "S" or "P"), or a string specifying a new extended channel.

The exact form of the channel specifier is documented separately for each new channel provided, but generally it consists of a letter followed by a ">" character; optionally, there may be further parameters inside the string (usually separated by commas). If you want to calculate a value for a parameter, this is perfectly possible - just remember to use the STR$ function to build up the string.

For example, to open stream 5 to a region of memory specified by the variables *addr* and *len*, use the following command:

```
OPEN #5, "m>" + STR$ addr + "," + STR$ len
```

CLOSE #*stream*

This command closes a stream, flushing any output that might be buffered. If the stream was one of the standard streams (0-3), then it is re-assigned to its default channel (ie "K" for streams 0 & 1, "S" for stream 2 and "P" for stream 3).

For example, to close stream 5:

```
CLOSE #5
```

FN ptr#(*stream*)

This function returns the current position of the the pointer in the stream specified. Not all channels support this function; see the documentation for individual channels.

For example, to find the position of the pointer in stream 5:

```
LET pos=FN ptr#(5)
```

FN ext#(*stream*)

This function returns the extent (size) of the stream specified. Not all channels support this function; see the documentation for individual channels.

For example, to find the size of stream 5:

```
LET size=FN ext#(5)
```

POINT #*stream*, *pos*

This command sets the current position of the pointer for the stream specified. Not all channels support this function; see the documentation for individual channels.

For example, to set the pointer to the start of stream 5:

```
POINT #5, 0
```

Packages

Starting with v2.00, it is possible to load a number of optional packages into your interface's memory, which extend the capabilities of *ResiDOS*. To install or uninstall a package, download it from this page and use the **%INSTALL** and **%UNINSTALL** commands. Alternatively, you can download the following BASIC program which guides you through the process of installing or upgrading a package:

- pkginst.tzx - Package installer (tape-loadable version)
- pkginst.bas - Package installer (version to copy direct to flashdrive)

All packages are provided here in two formats:

.tzx
: A tape file that can be loaded into memory before installation (or saving to flashdrive)

.pkg
: A file that can be copied straight to a FAT-formatted flashdrive, and installed directly from there. To do this, you'll need a version of the FATfs package already installed.

Available packages

The following packages are currently available:

- Channels, adds channels for accessing disk files, memory, variables and extremely flexible text windows.
- FATfs, adds support for FAT-formatted flashdrive

- TapeIO, adds support for LOADing and SAVEing to .TAP and .TZX emulator files
- TaskMan, allows saving snapshots and switching between multiple tasks
- ZX80, a ZX80 emulator
- ZX81, a ZX81 emulator

The ResiDOS Task Manager

The Task Manager

The *Task Manager* is an optional package that you can install on *ResiDOS*. Note that the current version (v1.00) only operates with 48K tasks, so you cannot save snapshots of 128K games. You should also only attempt to run one 128K game as a task at a time, otherwise the extra Spectrum memory will be overwritten by the 2nd 128K game. This limitation will be removed in a future release.

When you press the NMI button on your interface, the Spectrum immediately stops whatever it was doing (whether you were in BASIC, a word processor, a commercial game or whatever) and enters the *Task Manager*.

The *Task Manager* screen shows a list of the current tasks running on your Spectrum, together with the menu options that are available to you. When you first enter the task manager, the task list will consist of just a single task, named "Task 0", which is whatever you were doing at the time. To return to it, simply press ENTER.

Using the RAM in your interface, you can store several tasks at the same time, and switch between them instantly from the task manager screen. The number of tasks you can have running at once depends on the amount of RAM in your interface. The maximum number of tasks allowable is 16 (this is only possible on a 1024KB **ZXCF range** or **ZXMATRIX** interface).

The menu options described below allow you to manage your tasks quickly and easily.

Managing Tasks

The options available are shown on the task manager screen. Simply press the highlighted letter to select the option.

Cursor keys (up and down)

Use these to move the highlight up and down the task list.

ENTER

Re-enters the highlighted task.

New

Creates a new task, activates it and resets the Spectrum into BASIC. (Don't worry, all the other tasks are still there!)

Copy

Makes a copy of the highlighted task.

Delete

Deletes the highlighted task.

Purge

Deletes all tasks *except* the highlighted task.

Rename

Renames the highlighted task. Task names are up to 16 characters long.

Border

Sets the border colour for the highlighted task. Since it isn't possible for *ResiDOS* to know the border colour, it makes a guess which is sometimes wrong. You can change that here before saving a snapshot.

Snapshot

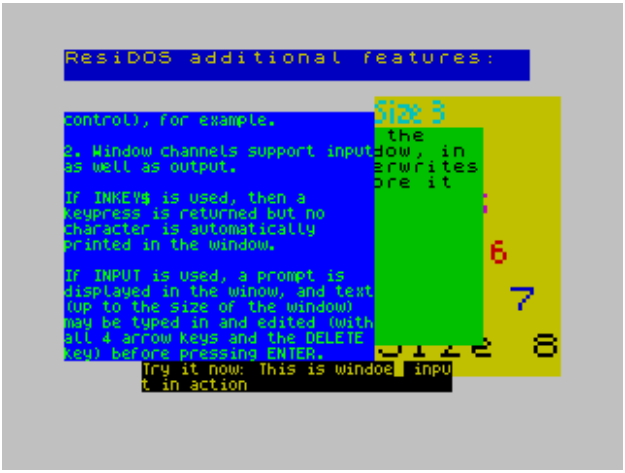
Saves a snapshot of the highlighted task onto the flashdrive in the standard .Z80 file format. This can be reloaded later using the **%SNAPLOAD "filename"** command (which can load any 48K or 128K snapshot in .Z80 or .SNA format).

Other options

Turn off Spectrum

This takes you to a special "shutdown" screen, from which you can turn off the Spectrum in safety. When you switch back on, the *Task Manager* will start up again, and you will be able to continue exactly where you were, with all your tasks still running.

Channels Package



On software package is windemor demo !!! se it in action !!

The channels package provides a number of new channels for use under *ResiDOS*. Most of these are compatible with the extended channels of the ZX Spectrum +3e, although there are also several important additions.

After installing the package, in order to make use of the new channels, you will need to read the information on extended channels.

The channels provided are:

- File access ("I>", "O>" and "U>")
- String variables ("V>")
- Memory regions ("M>")
- Text windows ("W>")
- Null ("Z>")

File channels

Streams may be opened to any file in one of three different access modes: input (using the **"I>"** channel), output (using the **"O>"** channel) and update (using the **"U>"** channel).

In input and update modes, the file must already exist, or an error will occur. In output mode, the file is deleted if it already exists, and then a new file is created. Once the file is open, it can be input from (if opened in input or update modes) or output to (if opened in output or update modes). Because of file buffering, it is essential to **CLOSE #** any file channel before finishing, otherwise data may be lost.

The channel specifier for all file channel types requires the filename of the file to follow the ">" character.

File channels support all the pointer operations (FN ptr#(), FN ext#() and POINT #).

It should be noted that files on IDEDOS disks are always stored as a number of 128-byte "records" and so you may read rubbish at the end of a file if it is not an exact multiple of 128 bytes in length. To avoid this problem, you should write some code or "signature" at the end of your file which you can detect when reading it back. Alternatively, you could write the number of bytes or entries in your file at the start. This problem does not exist for files on FAT disks, but it would be good practice to do this anyway, since it will then allow your program to be used with files on both FAT and IDEDOS disks.

Examples

OPEN #4, "o>test.txt"
Creates a file named "test.txt" on the default drive for output through stream 4.
OPEN #5, "i>a:test2.txt"
Opens a file named "test2.txt" on drive A: for input through stream 5.
OPEN #6, "u>c:/data/list.txt"
Opens a file named "list.txt" in the "data" directory of drive C: for update through stream 6.

Variable channels

The **"V>"** channel can be used to direct output to or input from a string variable, which can be easily manipulated within a BASIC program. This would allow you to (for example) examine disk catalogs in your BASIC program, or make an auto-running game demo (by inputting from a string containing set keystrokes).

The channel specifier for the variable channel type requires the name of the string variable to follow the ">" character.

Variable channels support all the pointer operations (FN ptr#(), FN ext#() and POINT #).

The string specified must be a character array with a single dimension, large enough to hold the maximum amount of data you expect to have to deal with.

Example

Here is an example program that outputs a list of installed ROMs to a string.

```
10 DIM a$(1000)
20 OPEN #8, "V>a$"
30 %roms+ #8
40 LET l=FN ptr#(8)
50 CLOSE #8
60 PRINT "Assignment length is ";l
70 PRINT "List is:"
80 PRINT a$( TO l)
```

Memory channels

The **"M>"** channel can be used in a very similar way to the variable channels. However, as it is a fixed memory region, it is more suitable for use by machine-code programs.

The channel specifier for the memory channel type requires the address and length of the memory region (separated by a comma) to follow the ">" character.

Memory channels support all the pointer operations (FN ptr#(), FN ext#() and POINT #).

Example

Here is an example program that outputs a disk catalogue to memory and then runs a machine-code routine to process it:

```
10 CLEAR 29999
20 OPEN #7, "M>30000,1000"
30 CAT #7
40 LET x=USR myroutine
50 CLOSE #7
```

Window channels

The **"W>"** window channels are the most complex of the extended channels currently available on *ResiDOS*. Two different types of window, with very different behaviours, are available:

+3e-compatible windows
: These windows are the default type, and provide the same facilities as those provided by the ZX Spectrum +3e (except that, under *ResiDOS*, input is also supported). These windows support Spectrum-style control codes, so that you can use normal **PRINT** items such as **INK**, **PAPER** and **AT** (amongst others) in your **PRINT #** commands, they also support Spectrum tokens and graphics characters (so that you can use **LIST #** directly, for example).

VT52-compatible windows
: These windows are created when you specify a character set address of zero (see below for details). These windows broadly emulate a subset of the VT52 terminal, and so use completely different control codes to the +3e-compatible windows. They are intended for applications such as *CP/M* and internet application console windows. They support only the standard ASCII character set, not Spectrum tokens or graphics characters. Additionally, they do not support line input; only (non-echoed) character input is

provided.

The channel specifier for the window channel type requires the following values (separated by commas) to follow the ">" character: top line (0-23), leftmost column (0-31), height (1-24), width (1-32), and optionally character size (3-8) and character set address. If no character size is specified, the default is 8. If a character set address is given, then this is used instead of the built-in fonts; this allows you to use nice fonts such as those provided with art programs and adventure games. If the character set address is given as zero (0), then a VT52-compatible window is created instead of a +3e-compatible window.

Note that the character size has no bearing on the way the window is defined, but it does affect the number of "actual" columns you have available. For example, the following defines a window the size of the entire screen; but because a character size of 5 is specified, the number of characters that can be printed in the window at any time is 24x51:

```
OPEN #5,"w>0,0,24,32,5"
```

Window channels do not support the pointer operations. However, the "set position" operation (POINT #) can be used to change the region of memory to which window output is sent. Normally, this is the standard Spectrum screen address (16384), but it can be set to any 8K boundary. This is mostly useful for machine-code programmers, as it allows them to perform window output directly to the 128K Spectrum's alternate screen at 49152 in page 7, for example.

+3e-compatible windows

When **PRINT**ing to +3e-compatible windows, you can use many of the same control functions as you can with the normal screen. For example: **'** (apostrophe; start a new line), **,** (comma; start a new column), **TAB**, **AT**, **INK** , **PAPER**, **FLASH**, **BRIGHT** , **INVERSE**, **OVER**. Of these, only **AT** behaves slightly differently; it takes y to be a pixel line and x to be an actual character column.

When first defined, windows are in non-justified mode, but they can be set to be left-, full- or centre-justified. Note that in justified mode, some features and control codes cannot be accessed, so you may need to switch back to non-justified mode to use them.

Input is supported in +3e-compatible windows. If you use **INKEY$ #** , then the keyboard is simply scanned and a character returned without anything being output to the window. If you use **INPUT #**, then a cursor is added to the window at the current position. The user can then input any text desired, using the left and right arrows to move along the text input so far, or the up and down arrows to move to the start or end of the text. The DELETE key deletes the character to the left of the cursor, and the ENTER key completes the input. Depending upon memory available, the entire size of the window can be used in the input, although care is taken to ensure that no input character is ever scrolled off the top of the window. An absolute maximum of 255 characters is allowed in the input line.

A complete list of control codes follows; these codes can be sent to a window by **PRINT**ing them using the **CHR$** function. If a code is preceded by (j) then it will be ignored if issued in justified mode (however, their settings will still be taken into account; for example, you can justify double-width text, but you must set it before entering justified mode). If a code is preceded by (e) then it can only be used in justified mode if the "embedded codes" feature has been set (which requires more memory to be allocated for the channel).

0
Turn justification off

1
Turn justification on

2
Save current window contents

3
Restore saved window contents

4

Home cursor to top left

5

Home cursor to bottom left

(j) **6**

Tab to left or centre of window (PRINT comma)

7

Scroll window

(j) **8**

Move cursor left

(j) **9**

Move cursor right

10

Move cursor down

11

Move cursor up

(j) **12**

Delete character to left of cursor

13

Start new line (PRINT apostrophe)

14

Clear window to current attributes

15

Wash window with current attributes

(e) **16,** *n*

Set INK n (0-7)

(e) **17,** *n*

Set PAPER n (0-7)

(e) **18,** *n*

Set FLASH n (0-1)

(e) **19,** *n*

Set BRIGHT n (0-1)

(e) **20,** *n*

Set INVERSE n (0-1)

(e) **21,** *n*

Set OVER n (0-1)

(j) **22,** *y, x*

Set cursor to pixel line y, character size column x

(j) **23,** *n*

TAB to character size column n

(e) **24,** *n*

Set attributes n (0-255)

(j) **25,** *n*

If n=1, then characters 165-255 will be printed as UDGs, with data located at the end of the standard UDGs. If n=0, then characters 165-255 will be printed as BASIC keyword tokens (the default).

(e) **26,** *n*

Fill window with byte n. Attributes are affected, but not cursor position.

(e) **27,** *n*

Fill window with character n. Attributes and cursor position are affected.

(j) **28,** *n*

Set double width (n=1) or normal width (n=0).

(e) **29,** *n*

Set height n (0=normal, 1=double, 2=reduced, 3=double reduced)

30, *n*

Set justification mode (0=left, 1=centred, 2=full)

(j) **31,** *n*

Allow embedded codes in justified mode if n=1.

VT52-compatible windows

VT52-compatible windows behave very differently from +3e-compatible windows, and are most useful for applications such as *CP/M* and internet console applications. They do not support line input (**INKEY$#** and **INPUT #** are allowed, but are implemented as non-echoed character input) or non-ASCII characters (such as the Spectrum tokens and graphics characters). By default, these windows operate in a "non-wrapping" mode (when the end of a line is reached, text does not wrap automatically to the next line unless CR and LF codes are issued) and with a visible cursor, although these modes can be changed. The full list of control codes supported is as follows (other control codes are ignored):

7 (BEL)
: Sounds a beep.

8 (BS)
: Moves the cursor 1 character to the left and replaces the character under the cursor with a space. If wrapping is enabled and the cursor is already at the left, it moves to the rightmost position of the previous line (unless it is already on the top line).

10 (LF)
: Moves the cursor down 1 line, scrolling the window up if necessary.

13 (CR)
: Moves the cursor to the leftmost column of the current line.

27 (ESC)
: Introduces an escape sequence or a literal character.

The following escape sequences are interpreted. If an *ESC* character is followed by any character other than those in this list, then that character is output as a literal (without any further interpretation) directly to the window, and the escape sequence is terminated.

ESC A
: Moves the cursor up. If the cursor is already on the top row, it has no effect.

ESC B
: Moves the cursor down. If the cursor is already on the bottom row, it has no effect.

ESC C
: Moves the cursor right. If the cursor is already in the rightmost column, it has no effect.

ESC D
: Moves the cursor left. If the cursor is already in the leftmost column, it has no effect.

ESC E
: Clears the window. The cursor position is not affected.

ESC H
: Homes the cursor (ie moves the cursor to row 0, column 0).

ESC I
: (Reverse index). Moves the cursor up 1 line, scrolling the window down if necessary.

ESC J
: Erases all characters from the cursor (including the character under the cursor) to the end of the window. The cursor position is not affected.

ESC K
: Erases all characters from the cursor (including the character under the cursor) to the end of the line. The cursor position is not affected.

ESC L
: Inserts a line. The row with the cursor on it and all rows below are scrolled down 1 line. The cursor row is cleared. The cursor position is not affected.

ESC M
: Deletes the row with the cursor on it. All rows below are scrolled up 1 line. The bottom row is cleared. The cursor position is not affected.

ESC N
: Deletes the character under the cursor. All characters to the right of the cursor are moved 1 column to the left. The character at the end of the row is cleared. The cursor position is not affected.

ESC Y *r c*

Moves the cursor to the given position. *r* is the row number plus 32, and *c* is the column number plus 32. If the position is beyond the edge of the window, the cursor is moved to the edge of the window.

ESC b *c*

Sets the foreground (ink) colour. *c* can be either a Spectrum colour (0 to 7) or a colour value compatible with Locomotive's +3 CP/M implementation, as follows:

- 32, black
- 34 or 35, blue
- 40 or 44, red
- 42 or 47, magenta
- 64 or 80, green
- 66 or 83, cyan
- 72 or 92, yellow
- 74 or 95, white

ESC c *c*

Sets the background (paper) colour. *c* can be either a Spectrum colour (0 to 7) or a colour value compatible with Locomotive's +3 CP/M implementation, as follows:

- 32, black
- 34 or 35, blue
- 40 or 44, red
- 42 or 47, magenta
- 64 or 80, green
- 66 or 83, cyan
- 72 or 92, yellow
- 74 or 95, white

ESC d

Erases all characters from the start of the window up to the cursor (including the character under the cursor). The cursor position is not affected.

ESC e

Enables the cursor blob.

ESC f

Disables the cursor blob.

ESC j

Saves the cursor position.

ESC k

Restores the cursor position (as saved by *ESC j*).

ESC l

Erases all characters on the current line. The cursor position is not affected.

ESC o

Erases all characters from the start of the line up to the cursor (including the character under the cursor). The cursor position is not affected.

ESC p

Enables reverse video mode.

ESC q

Disables reverse video mode.

ESC r

Enables underline mode.

ESC u

Disables underline mode.

ESC v

Enables wrapping at the end of a line.

ESC w

Disables wrapping at the end of a line.

Null channels

The **"Z>"** channel simply discards all output that is sent to it. It always returns a CR character (ASCII 13) when it is read from (so the **INPUT #** command always returns an empty string). This can be handy if you wish to execute a command which produces output that you do not want.

The channel specifier for the null channel type is simply "Z>".

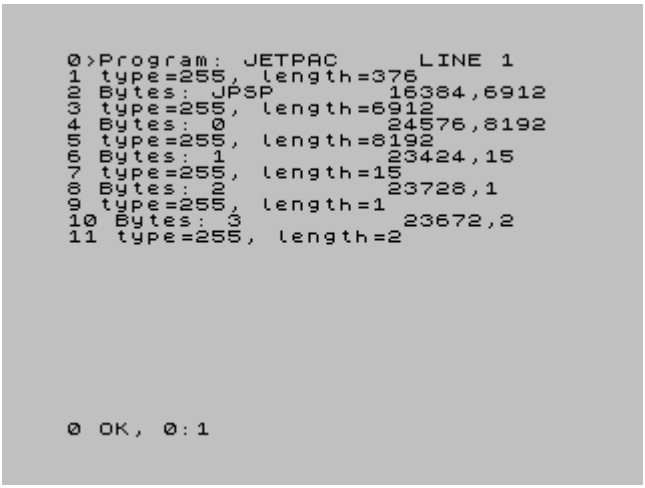
Null channels support all the pointer operations (FN ptr#(), FN ext#() and POINT #).

Example

Here is an example program that does nothing much at all!

```
10 OPEN #4, "Z>"
30 CAT #4
40 INPUT #4;x$
50 CLOSE #4
```

TapeIO Loading/saving from .TAP and .TZX files



TapeIO

The TapeIO package is an optional package that you can install on *ResiDOS* which gives you the ability to load and save directly to .TAP and .TZX files on your flashdrive as if it was a real tape. This means that you can take advantage of the thousands of .TAP and .TZX emulator files available and use them with your Spectrum

The most important commands to use are **%TAPEIN** and **%TAPEOUT**. Each of these takes the name of a .TAP or .TZX file. Thereafter, and standard **LOAD** commands that would normally access the tape will be redirected to whichever .TAP/.TZX file you specified in the **%TAPEIN** command. Similarly, any standard **SAVE** commands that would normally access the tape will be redirected to the .TAP/.TZX file specified in the **%TAPEOUT** command. In order to redirect access to tape, simply use the **%TAPEIN** or **%TAPEOUT**

command without a filename.

For example, to load up Jetpac from a *jetpac.tap* file, use the following commands:

- **%TAPEIN "jetpac.tap"**
- **LOAD ""**

To create a .TZX file containing your latest masterpiece, so that it can be loaded by an emulator, use commands such as:

- **%TAPEOUT "myprog.tzx"**
- **SAVE "My Program" LINE 10**
- **%TAPEOUT**

Controlling the tapepointer

Some more useful commands that are available are **%TAPELIST** and **%TAPEWIND**.

The first of these lists out the contents of the current .TAP/.TZX file being used for input (ie the last one specified in a **%TAPEIN** command). It also shows which is the next block that will be **LOAD**ed, with a cursor (similar to the cursor in a BASIC program listing).

With the **%TAPEWIND** command, you can "wind" the tape pointer forwards (using a positive number) or backwards (using a negative number) by a number of blocks, or to a specific block number (using the "=" option). If you do not specify any number, then the input tapefile will be rewound to the start.

For example, use the following commands to wind the tape in different ways and see the result on the tapelisting:

- **%TAPEWIND 3**
- **%TAPELIST**
- **%TAPEWIND -2**
- **%TAPELIST**
- **%TAPEWIND=6**
- **%TAPELIST**
- **%TAPEWIND**

Finally, you can also change the behaviour when the last file is loaded from the input .TAP/.TZX file. Normally, when this happens the tapefile is "ejected" and subsequent **LOAD** commands use the real tape again, until another **%TAPEIN** command is used. However, if you would prefer the tapefile to be "rewound" to the start, then use the following command:

- **%TAPEMODE 1**

To revert to the default behaviour (ejecting the tapefile), use:

- **%TAPEMODE 0**

Transferring files between disk and .TAP/.TZX files

It's also easy to transfer files from within .TAP and .TZX files to separate files on your flashdrive.
There are commands available to transfer files with headers (**%TAPEGET** and **%TAPEPUT**) and also headerless files (**%TAPEBGET** and **%TAPEBPUT**). For details, see the full command reference below.

Troubleshooting

All .TAP files should work correctly with *ResiDOS*, as well as some .TZX files which use the standard Spectrum loading system. However, any .TZX files using speedloading or other custom loaders will not work. If a game appears to stop loading from a .TZX file, then it is likely to have a custom loader. Using the **%TAPELIST** command on such a file should show which blocks *ResiDOS* can "see" and help to show if it is a non-standard loader program.

If you have a 128K Spectrum and find a game that fails to load, please try again after selecting 48K mode (with the **%SPECTRUM** command). Many games will only load in 48K mode by design (and would have included instructions on the cassette inlay to switch to 48K mode before loading).

TapeIO Command Reference

Here is a complete list of all the commands supported by the *TapeIO* package, including several options and variants that aren't described above. Please read carefully to get the most out of this package.

%TAPEIN "*filename*"
Redirect all tape LOAD commands to use the .TAP or .TZX file specified.

%TAPEIN
Redirect all tape LOAD commands back to physical tape.

%TAPEOUT "*filename*"
Create a new .TAP or .TZX file, and redirect all tape SAVE commands to it.

%TAPEOUT+ "*filename*"
Open an existing .TAP or .TZX file, and redirect all tape SAVE commands to it (appending to the end of the file).

%TAPEOUT
Redirect all tape SAVE commands back to physical tape.

%TAPELIST
List the contents of the current input tapefile.

%TAPELIST #*n*
List the contents of the current input tapefile to stream "n".

%TAPEWIND *n*
Wind the tapepointer by "n" blocks (positive is forwards, negative is backwards).

%TAPEWIND=*n*
Wind the tapepointer to block number "n".

%TAPEWIND
Rewind the tapepointer to the start of the tapefile.

%TAPEGET "*diskname*"
Copy the next file with a header from the input tapefile onto disk, using the specified name for the disk file.

%TAPEGET
Copy all files with a header from the input tapefile onto disk, translating names to fit in the 8.3 name format.

%TAPEBGET "*diskname*"
Copy the next block (could be a header or data block) from the input tapefile onto disk, using the specified name for the disk file.

%TAPEBGET+ "*diskname*"
Copy the next block (could be a header or data block) from the input tapefile onto disk, using the specified name for the disk file. Copy the entire block, including the block type and parity byte (which are normally stripped out).

%TAPEPUT "*diskname*"
Copy a disk file into the current output tapefile, complete with an appropriate header.

%TAPEBPUT "*diskname*"
Copy a disk file into the current output tapefile as a headerless block (of type 255).

%TAPEBPUT "*diskname*", *n*
Copy a disk file into the current output tapefile as a headerless block (of type "n").

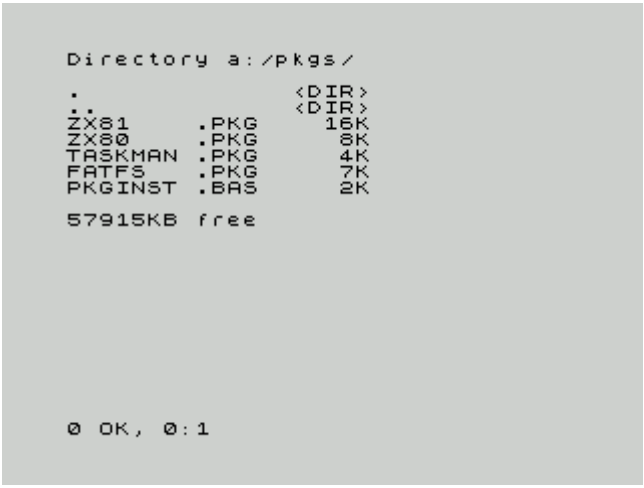
%TAPEBPUT- "*diskname*"
Copy a disk file into the current output tapefile as a headerless block, but don't add a type byte or parity byte.

%TAPEMODE *n*
Set the behaviour for what happens when the end of the input tapefile is reached (0=eject, 1=rewind).

%TAPEMODE+ *n*
Set the behaviour for what happens when the end of the input tapefile is reached (0=eject, 1=rewind). Make this behaviour permanent.

%TAPEMODE- *n*
Set the behaviour for what happens when the end of the input tapefile is reached (0=eject, 1=rewind). Remove any permanent behaviour.

FATfs



This package adds support for reading and writing to FAT16-formatted flashdrives to your *ResiDOS* system. This is incredibly useful, since all PCs and Macs can use this format of flashdrive. This means copying files (including other packages, upgrades to *ResiDOS*, snapshots and Infocom games for ZXZVM) between your PC and Speccy becomes extremely easy.

Once installed, any FAT16-formatted flashdrives can be accessed using *LOAD*, *SAVE* and all the other *ResiDOS* commands.

The current version has the following limitations, which will be addressed in a future release:

- no support for FAT32 or FAT12 flashdrives
- no support for long filenames
- partitions cannot be created, renamed or deleted by *ResiDOS*

Note that from *ResiDOS v2.01* this package is automatically installed together with the base system. You may, however, uninstall it at any time, or upgrade/reinstall it with the following downloads.

The *ZX80 emulator* is written by Paul Farrow, and allows you to emulate a ZX80 on your Spectrum. This package *requires a 128K Spectrum* since the normal Spectrum screen is used by the ZX80 for program data, and the alternate screen (only available in 128K Spectrums) must therefore be used. You will receive an "out of memory" error if trying to run in 48K mode.

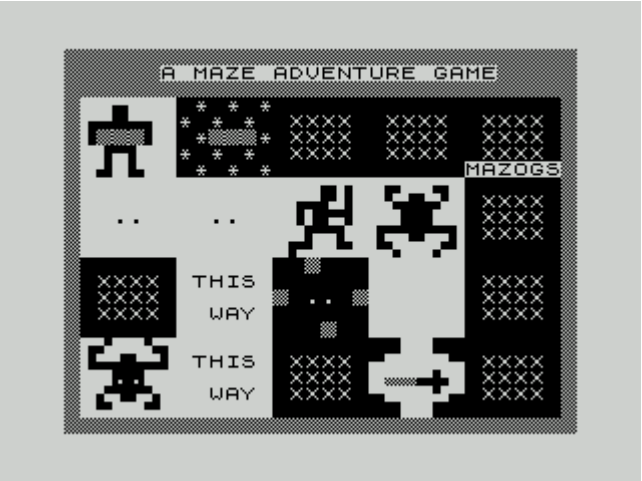
After installation, you can start the emulator with one of the following commands:

- **%zx80**
- **%zx80 "***snapshotfile***"**

The first command simply starts up the emulator, leaving you at the "K" prompt. The second starts the emulator and runs the named snapshot file, which must be in the standard ZX80 **.O** snapshot format.

It is also possible to LOAD and SAVE programs from within the emulator. When you enter LOAD or SAVE, a menu appears with several transfer options. Choose the *ResiDOS file* option to LOAD or SAVE to a file named **zx80.o** on the default flashdrive. It is not possible to specify other files from within the emulator, since the ZX80 does not use filenames. This means that after saving a file you are advised to exit the emulator (by resetting the Spectrum) and rename it for safe-keeping.

Full details of using the ZX80 emulator, including the option of copying programs directly from tape to flashdrive, are on Paul Farrow's website. A useful source of ZX80 snapshots is this website.



The *ZX81 emulator* is written by Paul Farrow, and allows you to emulate a ZX81 on your Spectrum. This package *requires a 128K Spectrum* since the normal Spectrum screen is used by the ZX81 for program data, and the alternate screen (only available in 128K Spectrums) must therefore be used. You will receive an "out of memory" error if trying to run in 48K mode.

After installation, you can start the emulator with one of the following commands:

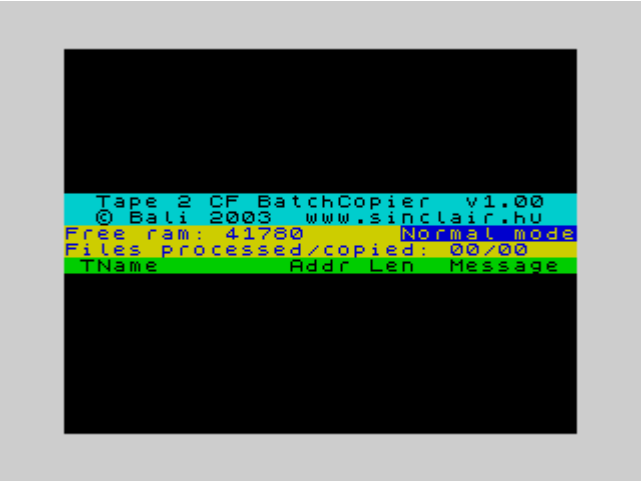
- **%zx81**
- **%zx81 "***snapshotfile***"**

The first command simply starts up the emulator, leaving you at the "K" prompt. The second starts the emulator and runs the named snapshot file, which must be in the standard ZX81 **.P** snapshot format. All snapshots should work, including high-resolution games.

It is also possible to LOAD and SAVE programs from within the emulator. When you enter LOAD or SAVE, a menu appears with several transfer options. Choose the *ResiDOS file* option to LOAD or SAVE to flashdrive.The file will be saved or loaded with the name given in the ZX81 LOAD or SAVE command, which may include drive letters, user areas and paths. It is advisable to save files with the **.P** extension, although this is not required.

Full details of using the ZX81 emulator, including the option of copying programs directly from tape to flashdrive, are on Paul Farrow's website. A useful source of ZX81 snapshots is this website.

Tape2CF



Written by *Bali*, this excellent program allows you to copy tape-based programs directly onto your flashdrive. The process is largely automatic, although you will need to modify BASIC loader programs after the transfer, so that they access the flashdrive rather than cassette.

TapeFile

This utility allows you to transfer files of any size between your PC/Mac and your Spectrum, using cassette tape for the transfer. It could also be used simply to backup programs from your flashdrive onto cassette.

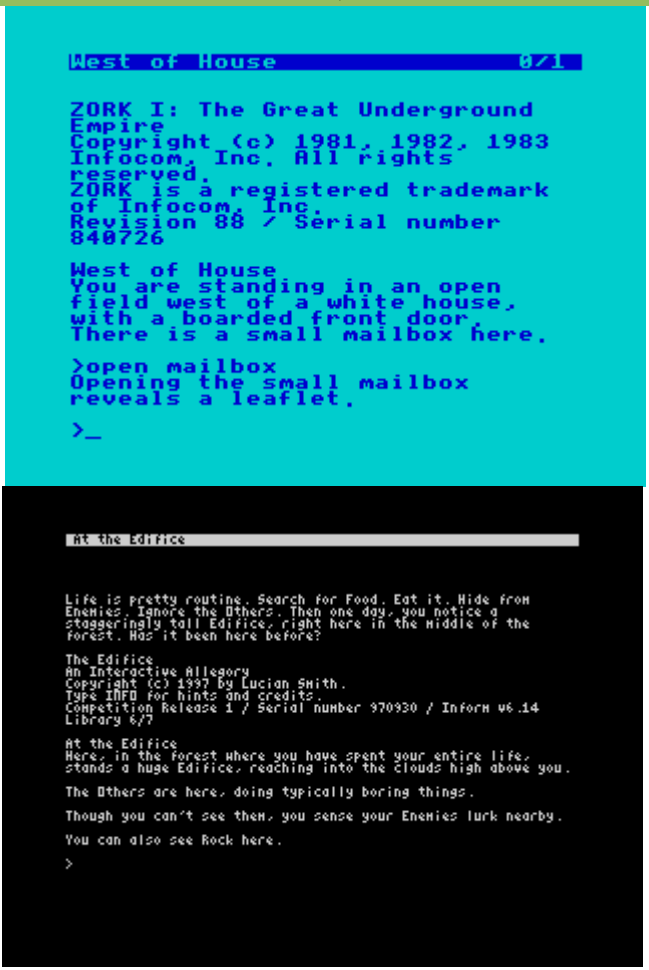
To use the program, you will need an emulator which supports the Interface 1 and can re-direct RS232 input and output to files. Such emulators include EightyOne (for Windows), Z80 (for DOS/Windows) and Fuse (for Unix/Mac OS X; note that at the time of writing, this facility is only available if you're prepared to download and build the emulator from the latest CVS sources).

To transfer a file between PC/Mac and tape, start the emulator and enable Interface 1 support. Then load the "tfe.tap" file in and follow the instructions.

On your Spectrum, load the "tfr.tap" file in; this will install itself to your flashdrive (thereafter you can just load the "tapefile.bas" program). This works in a similar way to the emulator side.

tfe.tap : TapeFile (emulator end)	tfe.tzx : TapeFile (emulator end)
tfr.tap : TapeFile (ResiDOS end)	tfr.tzx : TapeFile (ResiDOS end)

ZXZVM



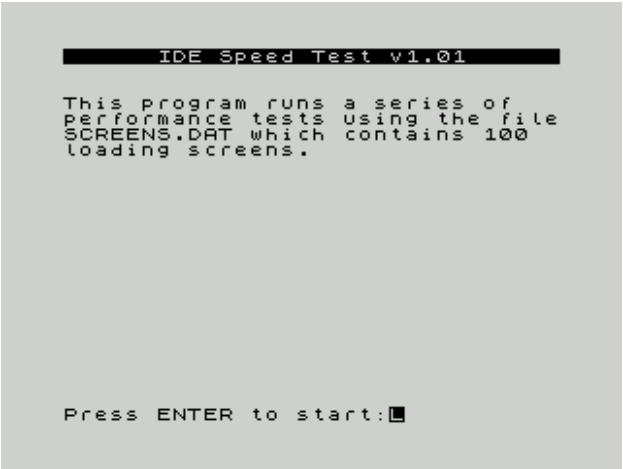
John Elliott's excellent ZXZVM, originally available for the +3/+3e and PCW series, has now also been ported to ResiDOS. It enables you to play Infocom's superb interactive fiction games, as well as hundreds of modern games written over the last few years. This port allows you to choose your preferred colours and character set, as well as to send a transcript to your printer and save/load positions to flashdrive.

The installation program guides you through your options. Once installed, load it with LOAD %"resizvm.bas" You can use *TapeFile* to transfer some suitable games from your PC/Mac. Hundreds of free games can be downloaded; the best place to start looking is at the Interactive Fiction Archive. ZXZVM can play games in the "zcode" format (sometimes known as z-machine, Inform or Infocom), and is capable of playing version 3, 4, 5, 7

and 8 games. This includes all the Infocom games except the last few graphical games (only a few of these can be downloaded free, however; Activision still sells compilation CDs of the rest).

- ZXZVM (includes documentation)
- ZXZVM source (for those interested!)
- IF archive for game downloads

IDESPEED



This program tests the data transfer speed of your hard disk system. Several tests are performed, using a data file containing 100 screen images. Transfer speeds are checked from the filesystem and swap partitions.

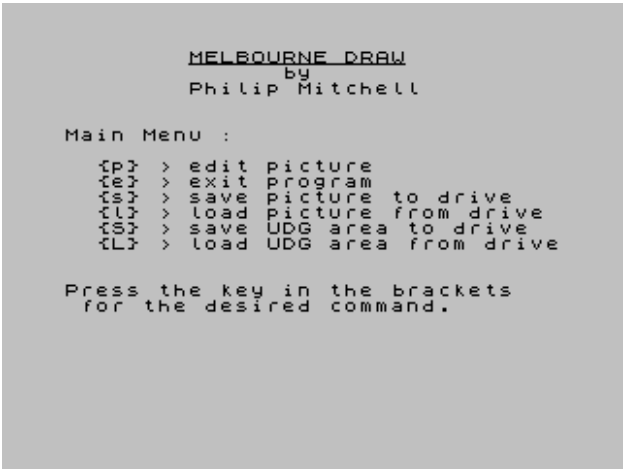
Also note that the transfer speeds are highly dependent upon the filesystem being used (the filesystem which is tested is the one to which you have copied the screens.dat test file). Currently, the FAT filesystem on ResiDOS is much slower than the IDEDOS filesystem (available on both +3e and ResiDOS).

If you have a swap partition, transfer tests will be performed using this as well; this will give higher speeds than either of the filesystems.

This program is available on the hard disk images (at the top of this page) due to the 675K file containing the screen data. ResiDOS users can also simply copy the files below onto a FAT disk for transfer purposes:

- idespeed.bas
- idespeed.c
- screens.dat

Melbourne Draw

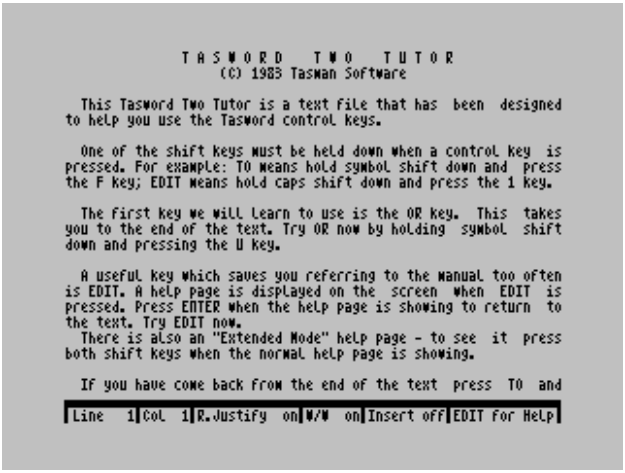


The classic drawing application, patched to save & load to a ResiDOS drive.

The following files can be copied direct to your ResiDOS drive: mdraw.bas mdraw.c

Full instructions may be found in the WOS archive: Melbourne Draw on Sinclair Infoseek

Tasword 2



The classic word processor, patched to save & load to a ResiDOS drive.

The following files can be copied direct to your ResiDOS drive: tasword.bas tasword.c tutor.txt

Full instructions may be found in the WOS archive: Tasword 2 on Sinclair Infoseek

Different ways of using ZXCF range, ZXUSB

Softrom Interface

1024KB / 16 gives 64 pages to play with OUT 4278, page (0-31) + 64 (enable memory write) + 128 (write to sram / read Spectrum ROM). This last bit 7 (128) turns of reading of sram but if you write at same address 0-16k it actually writes to sram(bit 6 needs to be set), in this way you can transfer code to sram by Spectrum basic.
OUT 4278, page (0-31) and sram is switched in.
Spectrum always resets to bank 0 with memory write protection on.
Other pages can be paged in and run by m/c.
(Above adress 4278 is for ZXCF, ZXUSB use 63)

Interface II

Sinclair game ROMS are loaded in by %INSTALL. Now you can actually have several roms installed and switch between them.

Transfer files from Interface 1, Microdrive

ZXMATRIX or ZXCF needs to be plugged after Interface 1 if this is to work.
Boot to ResiDOS.
LOAD from ZXMATRIX or ZXCF if you are to save something to Microdrive
OUT 4287,128 disables ResiDOS
LOAD/SAVE to Microdrive
OUT 4287,64
LOAD/SAVE to ZXMATRIX or ZXCF, (adress 4278 is for ZXCF, ZXUSB use 63)

Troubleshooting

The edge connector is a cut **ISA-16, it's not 100% Spectrum**, Check that there is **no cracks** on the soldering, if so re solder it. On some Spectrums you can't have it inserted all in, pull out some 4mm, **in some cases this is seen on the 5V rail on the interface, it drops to appr. 2.5V**. If interface can't keep memory on power off, **check battery it should not be less than 3V** when it's mounted on the interface(Spectrum turned off) if so replace it. If you are having **load/saving problems then either your flashdrive is broken or it's incompatible with the interface**. I recommend using SanDisk brand CF cards, from regular 1x speed to Extreme III 133x speed runs fine, you won't get any benefits on the faster CF card ;-)

If you having problems during installation, not finding all memory, problems with CF cards R/W error, problems with USB then you probably have a fault M1 line on your Z80 cpu, exchange cpu, only applies for ZXCF+, ZXCF+2, ZXMATRIX.

When installing ResiDOS, NMI has no function, by installing Taskman package you get some new functions like snapshot games and apps.
Latest ResiDOS install includes FATfs so you can access FAT formated flashdrive.
TapeIO package allows you to LOAD and SAVE to .TAP and .TZX emulator files.

Type %ROMS it shows all packages installed and at what page in the ZXMATRIX's sram chip.

ZXMATRIX

There are some jumpers, buttons, switches on the interface depending on what configuration you got it. The jumper left to the battery is removed when +2upg. Module is mounted.

RST = RESET, **NMI** = Interrupt, **U C** = USB / CF mode, **DB** = USB debug mode

UPL = Upload, this is used when installing ResiDOS on the interface , put jumper in or switch to Upload and reset the spectrum and load ResiDOS.

ZXCF Technical notes

CF card is working in 'mem-mode' by a 8bit data bus on I/O *191*. On board memory is supported by a CR2032 backup battery that prevents content be lost when Spectrum is switched off. Resides is uploaded to sram and is paged in on first 16k of memory space of Z80 (rom is switched out). Paging is done on I/O *4287d*. Bit 0..5 memory pages (max 1024KB), Bit 6 *unset* protects memory, Bit 7 *set* memory is switched off.

I/O #10BF (4287d) memory control

#00BF - #0FBF (191d – 1983d) CF control

#07BF (1983d) Status / Command
#06BF (1727d) LBA 24-27/drive
#05BF (1471d) LBA 16-23
#04BF (1215d) LBA 8-15
#03BF (959d) LBA 0-7
#02BF (703d) Sector count
#01BF (447d) Error / Feature
#00BF (191d) data buffer
#xx9F (159d) fast data buffer (ZXCF+2 only feature)

#07BF (1983d) Command register (wr)
Minimum required commands

Identify drive #EC (236d)
Device information

Read sector #20 (32d)
Read 512 bytes

Write sector #30 (48d)
Write 512 bytes

Request sence #03 (3d)
Gives extended error code after normal error

Execute drive diagnostics #90 (144d)
This performed when the device is first powered on.

Example code in basic for ZXCF

```
Dim p(512)
Rem 512 bytes of data to play with
Data 109,121,95,110,97,109,101,95,105,115,95,
115,112,101,99,116,114,117,109,32,49,50,51,52,
53,54,55,56,57,32,32,36
Restore
For a=1 to 512
Read d
Let p(a)=d
If d=36 then restore
Next a
If In 1983>127 then stop : Rem check if drive ready
Out 1727,160 : rem set head (LBA 24-27)/drive
Out 1471,0 : rem cylinder high (LBA 16-23)
Out 1215,15 : rem cylinder low (LBA 8-15)
Out 959,1 : start sector (LBA 0-7)
Out 703,1 : how many sectors
Out 1983,48 : Rem set to write 512 bytes
For a=1 to 512
Out 191,p(a)
next a
Out 1983,32 : Rem set to read 512 bytes
For a=1 to 512
Let p(a)=In 191
next a
For a=1 to 512 : Rem show contest on screen
Print Chr$ p(a);
Next a
```

Note that you might corrupt the contents of your CF card and might need to reformat it.

+2 upg. Second slot Tecnical notes

I/O 31d **Write** is the drive/cf slot select. 0=first 1=second drive.

ZXUSB Tecnical notes

I/O 127d **R/W** data buffer
I/O 63d **Read** bit 0 RXF, bit 7 TXE. **Write** Memory bank control
RXF = 1 do not read data buffer TXE = 1 do not write data buffer
Memory bank control bits works same as on ZXCF

ZXUSB has 2 USB slots, Slot 2 is used for single USBdrive it can be a USB adaptor with M2, miniSD, SD, CF, MMC, Memorystick and many more.. you can use USBmemory sticks, probably USB HDD (not tested). USB slot 1, this is possible to be used for pheripal's , this can be various things like a keyboard, mouse, printer, floppydrive, ethernet card, webcam you name it. Only flawn here is the software, we are at the begining .... after some months we probably should see some use of the slot.

Hardware requirements:

ZX Spectrum 48k, 48k+
ZX Spectrum 128k, +2
ZX Spectrum 128k +2A, +3

Screenshot not accurate

On my homepage there is a file ' software.zip ', unzip it and put files on a FAT16 formatted flashdrive. To launch it type ' LOAD %"run" '. You will find most if not all apps covered in this manual + more.

ResiDOS installation for all my interfaces + all package files.

There is also a folder 'MP3' and you will find ResiDOS installation in MP3 format for all my interfaces.
Now you can easily load ResiDOS to your interface with a MP3 player, computer, phone.

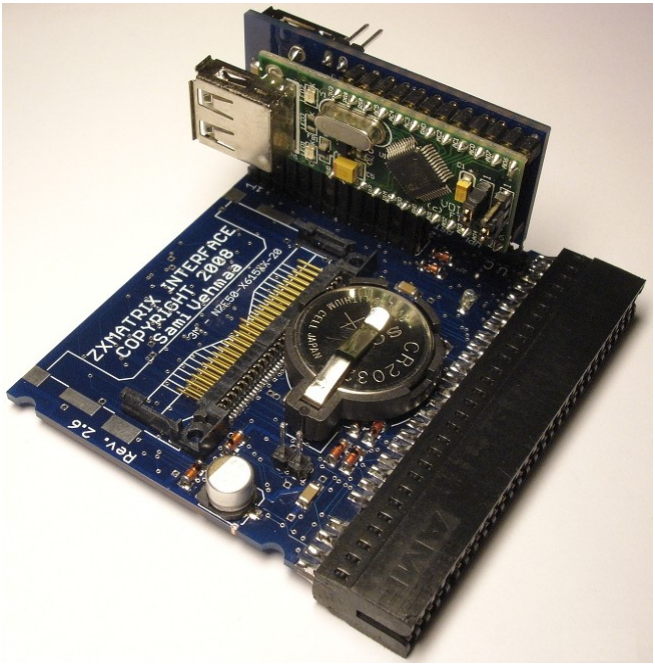
There is also a BETA version of DEVICES.pkg with demo app.
If you are into coding USB devices this might be some fun to you. Ohh it's for USB mice & keyboard.
No support is given, if you can't make it run, don't ask it's beta, ms mice works fine.

Some final words:

Garry and me have put many month's/hours of work on this interface and I'm looking forward seeing all things that can be made with this interface and there is practically no end.... there are amounts of stuff that can be connected and written software for.

Interfaces I have built so far

ZXATASP	2002	IDE /CF/ 512KB sram
Jessa I/O	2002	Serial interface 0.5mbit
ZXCF	2003	CF / 512 or 1024KB sram
ZXCFk	2007	CF / 512KB sram / kempston joystick
ZXCF+	2007	CF / 512 or 1024KB sram / boxed
Kempston module.	2008	Kempston Joystick module for ZXCF+
ZXCF+ /2 upg.	2008	Kempston & second CF slot
ZXMATRIX	2008	Dual interface ZXCF+2/ZXUSB



EOF