ElkSD128

SD Card Interface, 128K RAM Expansion and Joystick Interface for the Acorn Electron

---

www.ramtop-retro.uk

User Guide

Last updated 26/11/2020

1. Overview

The ElkSD128 is a versatile, multi-function interface cartridge for the Acorn Electron that provides many features, including:

- A fast SD card based storage interface compatible with commonly available .SSD disk image files.
- 128KB of ‘sideways’ RAM, expanding the Electron to a total of 160KB of memory.
- A 9-pin Atari/Commodore type joystick port supporting three interface systems plus keyboard emulation.
- SD card operations use the popular open-source MMFS file-system for maximum compatibility.

This manual serves as both a guide to using the ElkSD128 and a reference for directly programming the hardware. The early sections focus on using the interface, with technical and background information given later.

2. Installing the Interface

Your ElkSD128 interface should be packed in an anti-static bag or small cardboard box. Remove the interface from the bag or box and check that it is intact and has not been damaged in shipping.

To install the interface, follow these steps:

1. Clean the Electron's expansion connector, preferably with isopropyl alcohol or contact cleaner solution. If these are not available rub the connector with a pencil eraser.
2. Connect the ElkSD128 to the expansion connector. It will only fit one way. *Make sure the Electron is switched off before doing this!*
3. Insert your joystick into the 9-pin port on the interface. Ensure the joystick is of a compatible type (see section 4).
4. Switch on the Electron.
5. If the Electron fails to start or the screen is distorted, switch off and clean the expansion connector again.

3. Using SD Card Storage

The SD card slot on the ElkSD128 enables the saving and loading of programs using a system of 'disk images'. Up to 512 disk images may be contained on one SD card.

It is recommended that you use an SD card with 1-8GB of capacity and formatted as FAT32. Cards larger than 8GB will often work but this is not guaranteed, and such cards may need reformatting to FAT32.

In the Acorn world most existing disk images are in .SSD format. For the ElkSD128 to use these they must be bundled up into a master file called 'beeb.mmb' that is copied to the SD card. Details on how to so this are given in section 6.

To get started, you can download a pre-made beeb.mmb file containing many Electron games from:

Unzip this file and copy beeb.mmb to the SD card, insert it into the ElkSD128 and switch on.

Hold down the SHIFT key and tap BREAK and the Electron will reboot and load a menu system, permitting easy access to the games stored on the SD card.

The beeb.mmb files contains many disk images and a range of commands area available for managing and using disk images.

Each disk image is stored in a logical ‘slot’, identified by a number. The first disk image is slot 0, the second slot 1, etc.

Typing

```
*DCAT
```

Will give a list of disk images on the SD card, including their slot number.

Any given disk image can be ‘inserted’ – made available for use – by typing the command *DIN followed by the slot number. So, typing:

```
*DIN 3
```

will load the *fourth* disk image (fourth because the first slot is zero, not one). To see a list of files on the image you can type:

```
*CAT
```

The normal SAVE and LOAD commands work with MMFS just as they do with a tape drive or DFS disk system. Lots more information on MMFS commands can be found in the MMFS reference at the end of this guide, as well as a link to extensive documentation on the MMFS Wiki pages.

4. Using the Joystick

The ElkSD128 joystick port is electrically compatible with most 9-pin Atari/Commodore type joysticks. Sticks designed to work with a wide range of systems – C64, Atari 8-bit, Amiga, ST, etc – should function with issue.

Game pads are not supported and connecting them is not recommended. Also, any device using analogue signals such as paddles, mice, trackballs, etc, will not work.

Joysticks with a grey 9-pin plug intended for use with the Spectrum +2 will not work without an adapter.

From a software perspective, joystick support on the Electron is not entirely straightforward as the machine was not designed with joystick use in mind. There are a variety of interface standards supported by some games, but the majority of Electron games have no built-in joystick support at all.

The ElkSD128 features four methods of joystick support:

1. Acorn Plus 1 joystick port
2. First Byte interface
3. Slogger interface
4. Keyboard emulation

Acorn's Plus 1 expansion unit provided a single analogue type joystick port, compatible with that found on the BBC micro. If a game supports the Plus 1 joystick port no configuration is generally necessary to use it with the ElkSD128. Select the joystick option in the game (some games will prompt you to press fire to detect the joystick) and it will work.

The First Byte and Slogger interfaces are devices that added a digital joystick port to the Electron. Games may prompt you to enter the I/O address of the joystick interface, which on the ElkSD128 is &FCC0 for the First Byte and &FCD0 for the Slogger. Games may also ask for a ‘code’ to be entered, being either 0 or 1 – this determines if the joystick signals are active high or low at a hardware level. Try both, one should work correctly.

Finally the ElkSD128 offers a keyboard emulation system. This detects joystick movements and generates a pseudo key-press so games with only keyboard control can use the joystick. Many games will work with this system, but not all.

After plugging in the joystick for the first time we recommend you run the joystick test utility to check it is working correctly.

This can be done with the command:

*JOYSTICK 2 TEST

(Which can be abbreviated to *J. 2 T. for brevity)

Configuring the keyboard emulation system can be done by typing:

*JOYSTICK 2 SETUP

(or *J. 2 S.)

You will be prompted to enter keys for UP, DOWN, LEFT, RIGHT and FIRE, and also an ‘address’. This is where the small machine code program that enables keyboard emulation resides in memory. The default address is &150 – you can press enter to accept the default.

After this you can start your game. Note that you must boot the game by entering a command rather than pressing SHIFT-BREAK or the keyboard emulation will not work.

For games stored on SD card this is usually done using the ***DBOOT** command, followed by the 'slot' number of the game. If you have downloaded the games collection from the ElkSD128 web site, you can boot the main menu using ***DBOOT 0**

Technical details of how the ElkSD128 deals with joystick support and information on using the joystick in your own programs can be found in section 9.

5. Sideways Memory

The 6502 processor chip used by the Electron can only support up to 64K of memory at once. By default the Electron divides this into 32K of RAM for user programs and screen data, 16K for the ROM holding BBC Basic, and finally 16K ROM for the machine operating system (MOS).

However, Acorn recognised the requirement for a way of adding more memory to the computer and designed the 'sideways memory' system that enables the BASIC ROM to the temporarily disabled and bank of additional RAM or ROM made available in its place.

The Electron supports 16 sideways banks, each 16K in size. Four of these (banks 8,9,10 and 11) are reserved for use by the Electron system for various purposes.

When the ElkSD128 is fitted, banks 12, 14 and 15 are occupied by ROMs which enable SD card and joystick support. Banks 0-

7 are mapped as RAM, giving the Electron an additional 128K of RAM, for a total of 160K.

There is one limitation to note, however; the MMFS software used to provide SD card functionality requires one 16K bank of RAM for its own use, so bank 7 is not available for user programs. At boot time bank 7 is locked to read-only access and will appear as a ROM bank.

(**note**: the locking of bank 7 is why the ElkSD128 displays 144K at boot up, not 160K. Unlocking the bank will cause the RAM count to increase to 160K.)

The ElkSD128 contains the RH Plus 1 utilities ROM in bank 15. This provides the code necessary to support Plus 1 joystick operation, but also contains many useful commands for using sideways memory. See section 10.

6. Creating a Beeb.mmb File

To create your own beeb.mmb file or to add .SSD files to an existing one you will need to download the MMBImager software from

https://github.com/dandelion-labs/MMBImager

If you are creating a new beeb.mmb file, click '**File'** then '**New Image'**, and select the root directory of your SD card as the destination. When prompted enter 'beeb' as the filename.

To use a pre-existing beeb.mmb, select 'File' then 'Open Image' and choose the beeb.mmb file you would like to use.

Select a disk image slot in the main window and click '**SSD Image'** then '**Load Image'** to insert an SSD image in to that slot.

When you have loaded all images, click '**File**' then '**Close Image**'.

Note: on some PCs MMBImager may not run due to an error that references the '**MSCOMCTL.OCX**' file. To fix this problem follow the steps below:

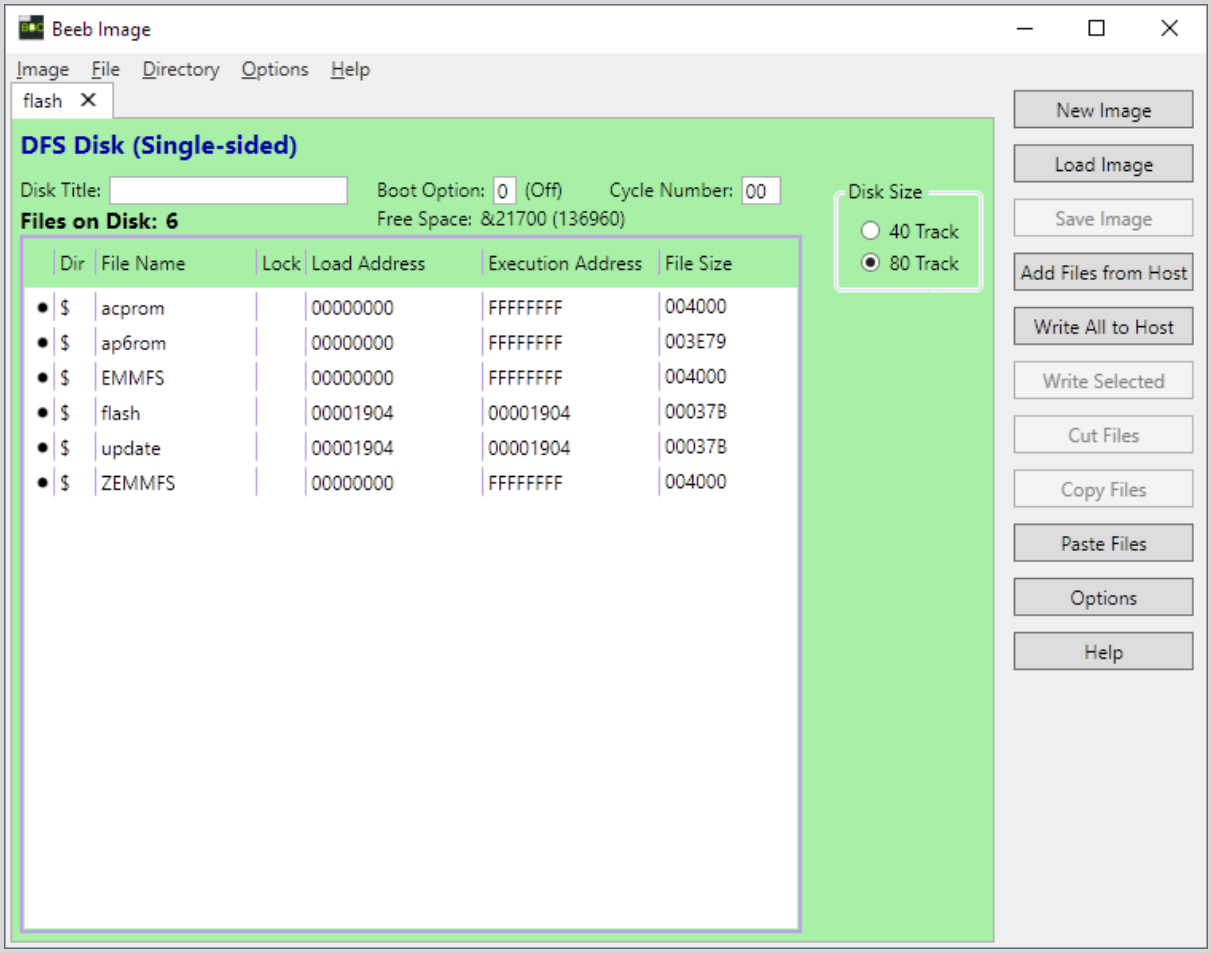
- Download the archive containing MSCOMCTL.OCX from Microsoft by clicking here : https://www.microsoft.com/en-us/download/details.aspx?id=10019
- Run the installer you just downloaded
- If you have a 64-bit Windows system, open a command prompt (CMD.EXE) and type: **copy c:\windows\system32\mscomctl.ocx c:\windows\syswow64**
- If you have a 32-bit Windows system, open a command prompt (CMD.EXE) and type: **copy c:\windows\system32\mscomctl.ocx c:\windows\syswow32**
- Then finally, type **: regsvr32 mscomctl.ocx**
- Reboot the PC

7. Using .SSD Files

To add your own files to an existing disk image, a program called Beeb Image is used. You can download a copy here:

http://www.cowsarenotpurple.co.uk/beeb-image.html

Open Beeb Image and click 'Load Image' to select the SSD file. Then click 'Add files from host' and select the files you wish to add. Finally, click the image menu and 'Save Image' to save the modified SSD.



8. Compatibility and Memory Use

On the ElkSD128 the MMFS filesystem software resides in sideways RAM. This is done to address a compatibility issue that crops up when using disk-type storage systems on the Electron.

No program can run only from ROM; it will need some RAM to store data. Acorn's DFS (Disk Filing System ROM) gets the RAM it needs by commandeering a small section of the Electron's main 32KB memory area. On 32KB Electrons MMFS works the same way, using around 2.5KB of main system RAM.

BBC Basic has a variable called PAGE, which lists the lowest byte of memory available for use by user programs. The higher PAGE is, the less memory is available. On an unexpanded Electron with just a cassette player attached PAGE will be set at byte 3845 ($E00 in hexadecimal). But with a disk system attached PAGE can rise to 6400 ($1900) or even 7424 ($1D00).

This poses no problems with games originally supplied on floppy disk, they expect DFS to use up some memory and make sure not to store anything there. But games written to load from tape will often expect to be able to use all of the Electron's 32KB, including the area reserved for DFS or MMFS.

Many of the Electron game disk images available for download have been converted straight from tape, and will just blindly trample all over the reserved memory area. The practical result is that loading such a game from a memory card will cause MMFS to crash or malfunction due to memory corruption. On a 32KB Electron there is nothing to be done about this other searching for a copy of the game that has been patched to work with PAGE at $1900.

However, there is a way to circumvent this problem when sideways memory is available. The ElkSD128 holds a copy of MMFS in ROM, which is copied to sideways RAM at boot time. This way a fully functioning copy of MMFS that uses no system RAM can be used without problems.

The first 13.5K of the sideways RAM bank used by MMFS (bank 7) is locked after MMFS is copied in, to prevent badly coded software from corrupting the filesystem. See section 9 for details of how the bank lock can be disabled.

9. Hardware Registers

The ElkSD128 implements several new hardware registers for accessing the joystick port, SD card interface and sideways memory system. In order they are:

&FC70 – Plus 1 analogue/digital converter. This register mimics the Plus 1 ADC chip; the required ADC channel is selected by writing a decimal value to this register, then reading back a value.

On a real Plus 1 A/D conversion takes time and bit 6 of &FC72 goes low when conversion is complete and valid data may be read from &FC70. This is not the case with the ElkSD128. As no actual A/D conversion happens valid data can be read from &FC70 immediately, regardless of &FC72 bit 6 status.

Immediately reading data from &FC70 will work on an ElkSD128 but not necessarily on a Plus1, so it is strongly recommended that code be tested on both the ElkSD128 and a real Plus 1.

&FC72 – Plus 1 joystick and ADC status. Bit 6 of this register going low is used by the plus 1 to indicate a completed A/D conversion. On the ElkSD128 bit 6 is always low. Bits 4 and 5 going low indicate first and second joystick fire buttons respectively have been pressed.

&FC80 – SPI controller data port. This is a byte-wide register that handles reading and writing to the SD card via SPI. Access to this register should take place only within the MMFS SPI driver or data corruption may occur.

&FC80 – SPI controller clock/status register. The least significant bit of this register indicates if the SPI controller is busy; 0 = idle, 1 = busy. Writing to the LSB controls the clock speed of the SPI interface, 0 = ‘slow clock’ for use during SD

initialisation, 1 = fast clock used after initialisation. Access to this register should take place only within the MMFS SPI driver or data corruption may occur.

&FC82 – Sideways RAM lock. When set to 0, bit 7 of this register causes the first 13.5KB of sideways bank 7 to be write-protected. Also, bit 6 can be set to 1 to enable write-protect on the first 13.5KB of bank 6 – banks 0-5 cannot be locked; this is due to lack of logic space in the ElkSD128's CPLD chip.

(note that the actions of bits 7 and 6 are inverted – this is because the register is initialised to 0 at power up, and so bank 7 is locked and bank 6 unlocked by default.)

&FC83 – Device ID. Always returns 128 (&80) for ElkSD128 rev 1.0.

&FCC0 – First Byte interface register. Bits 7-5 are always 1. Bits 4-0 will go low to indicate joystick fire, left, right, up or down.

&FCD0 – Slogger interface register. Bits 7-5 are always 1. Bits 4-0 will go high to indicate joystick fire, left, right, up or down.

10. Sideways Memory Commands

The RH Plus 1 ROM contains useful commands related to sideways memory. Some of them are given below:

***ROMS**

Lists sideways ROM and RAM details.

***SRLOAD <filename> <bank>**

Loads the named file into sideways RAM, ie, *SRLOAD “foo” F will load a file called ‘foo’ into bank 15 (F in hex).

*SRSAVE <filename> <bank>

Saves the contents of the specified bank to a file. Works on both ROM and RAM banks.

*SRWIPE <bank>

Erases the specified RAM bank.

*UNPLUG <bank>

Disables any sideways ROM in the specified bank.

*INSERT <bank>

Enables a ROM on the specified bank; needs BREAK for the OS to pick up the ROM.

11. MMFS Commands

The full MMFS command reference can be found at https://github.com/hoglet67/MMFS/wiki/Command-Reference

Parameters

- <drive> Drive number (0 to 3)

- <dno> Disk number (0 to a maximum of 511 dependent on the MMB image size)
- <dsp> Disk specification, i.e. disk title (not case sensitive)
- <adsp> Ambiguous disk specification, i.e. disk title terminated with optional wildcard “*”

Parameters in brackets are optional, and “/” indicates “either/or”.

Where the drive is optional, if no drive is given then the current drive (set using *DRIVE) is assumed.

Note: If disk titles contain spaces they must be enclosed in quotes.

Commands

*DBOOT

Syntax: *DBOOT <dno>/<dsp>

Inserts specified disk into drive 0 and boots it (if boot option of disk set).

Example:

>*DBOOT Planetoid

*DCAT

Syntax: DCAT ((<from dno>) <to dno>) (<adsp>)

Lists disks in disk number order.

An optional disk number range and/or ambiguous disk title can be specified.

This displays all (formatted) disks with numbers in the range 10 to 20 whose disk title begins with “R” or “r”.

Note: “P” after the disk title indicates that the disk is locked (read-only), and the number of disks found matching any specification is given at the bottom of the list.

*DDRIVE

Syntax: DDRIVE (<drive>)

Lists the drives and which disks are currently “inserted”, plus their status.

*DFREE

Syntax: DFREE

Displays the number of unformatted disks, and the total number of formatted and unformatted disks.

*DIN

Syntax: DIN (<drive>) <dno>/<dsp>

Insert specified disk into drive.

If the drive number is omitted, the current drive is used.

Note: A disk cannot be in more than one drive at a time. E.g. if Disk 200 is in Drive 0, and the user enters *DIN 2 200, Disk 200 will then be in Drive 2, and “No Disk” will be in Drive 0.

*DOP P

Syntax: DOP P (<drive>)

Protects (locks) the disk in a drive so that it is read-only.

If the drive number is omitted, the current drive is used.

*DOP U

Syntax: DOP U (<drive>)

Unprotects (unlocks) the disk in a drive so that it can be written to.

If the drive number is omitted, the current drive is used.

*DOP K

Syntax: DOP K (<drive>)

Marks the disk in a drive as unformatted.

Unless *ENABLE is used before this command, the user is asked for additional confirmation (i.e. twice)

If the drive number is omitted, the current drive is used.

NB: This command does not change any data on the disk “surface” and can be undone using *DOP R.

*DOP R

Syntax: DOP R (<drive>)

Marks a previously unformatted disk as in the drive as formatted.

If the drive number is omitted, the current drive is used.

*DOP N

Syntax: DOP N (<drive>)

Finds the first unformatted disk and places it in specified drive.

Note, the disk must then be formatted in the usual way.

*DRECAT

Syntax: DRECAT

Rebuilds the “Disk Table”. The disk table contains a copy of the disk title, plus the disk status (read-only, read/write, unformatted or invalid).

The title in the disk table is updated when *TITLE is used, or a (new) disk is formatted using *DOP N. However, copying a disk using *BACKUP etc. will not update the “Disk Table”. In this case *DRECAT can be used to refresh the disk table.

*DABOUT

Syntax: DABOUT

Prints the name of the original author of MMFS.

12. Acknowledgements

The ElkSD128 interface uses the MMFS software, originally developed by Martin Mather and maintained by David Banks, and the RH Plus 1 ROM which is maintained by J.G. Harston.

Many thanks to these gentlemen for their excellent software, on which the ElkSD128 depends.