ElkSD-Plus 1

Revision 2.x

Acorn Electron SD Card Interface & 48K RAM Expansion

www.ramtop-retro.uk

User Guide

Last updated 10/05/2020

1. Introduction

The ElkSD-Plus 1 cartridge is an expansion for the Acorn Electron and Acorn Plus 1 unit with the following features:

- Load and Save software using standard SD cards up to 8GB
- Inserts into a Plus 1 cartridge port
- Expands your Electron to 48K of RAM
- Enhanced compatibility, works with almost all Electron games
- Uses standard SSD disk image files
- Runs the popular MMFS filesystem
- Hardware SD controller for fast read/write speeds
- 3D printed protective case

The purpose of this guide is to explain the inner workings of the ElkSD-Plus 1 and effective use of the device. The guide assumes you have read the installation instructions included with the cartridge, it is installed and working, and you are familiar with using disk images. A copy of the installation instructions and other documentation is available at http://ramtop-retro.uk/elksdp1.html

2. How the SD Interface Works

For many 8-bit home micros from the 80s adding support for removable solid-state storage is as easy as attaching a floppy drive emulator, like the famous GoTEK, to the floppy drive port.

On Acorn's 8-bit systems floppy support has three components; the floppy drive mechanism itself, the drive controller chip, and a piece of system software called the *Disk Filing System* or DFS. Unfortunately, the Electron's budget origins mean it was shipped without any of these components included. Acorn provided them in the form of the Plus 3 disk peripheral, which is now rare and expensive to obtain.

As we want to use memory cards the floppy drive and controller chip are unnecessary. But we do need the DFS, or rather something that is compatible with it but actually talks to an SD card rather than a floppy drive. Such software exists in the form of MMFS, which is a DFS compatible package designed to work with memory cards. The ElkSD-Plus 1 interfaces a standard SD memory card slot to the Electron and provides a copy of MMFS stored on a flash chip which enables the Electron to communicate with the SD card. It also contains 32KB of RAM, half of which is used to expand the Electron to

48KB of memory via the ‘sideways RAM’ system. The other 16K of memory is used to enhance compatibility with games (detailed in section 5).

Revision 2.x of the ElkSD-Plus1 uses a new hardware based 8-bit SPI interface to drive the SD card, which results in up to 5x faster reads and 4x faster writes compared to the revision 1.x cartridge.

3. Understanding MMFS

MMFS works much like Acorn’s DFS, but because a memory card can store potentially hundreds of disk images there are some additional considerations to deal with.

The most common disk image format for Acorn machines is .SSD (Single Sided Disk) format. MMFS uses this format, but rather than having many files scattered on a memory card it bundles them all into a single file called *beeb.mmb*, which is pre-prepared and copied to the memory card.

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.

On the Electron 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. Going Sideways

To fully understand how the ElkSD-Plus 1 and MMFS work, it’s important to know about Acorn’s ‘sideways’ memory system.

The Electron uses the 6502 processor, which can only access a total of 64KB of memory at one time. Acorn decided to split this 64KB ‘address space’ into three areas; 32KB of RAM for storing the screen display and user program code, 16KB for a ROM chip containing the computer's system software (MOS or Machine Operating System in Acorn nomenclature), and the final 16KB for a ROM containing the BBC Basic language.

In the interests of expandability, Acorn gave the Electron the ability to temporarily 'page out' the BBC Basic ROM and replace it with either another ROM chip or 16KB of additional RAM. These are known as 'sideways' ROM and RAM respectively.

(it is worth pointing out here that just because the page containing BASIC and other sideways pages -- such as the one containing MMFS -- cannot be visible to the 6502 processor at the same time does not prevent using the SD card from BASIC. The Electron is smart enough to switch out BASIC during card I/O operations, when access to MMFS is required, and switch BASIC back in afterward. This happens seamlessly.)

A total of 16 sideways pages are supported by the Electron. Pages 8, 9, 10 and 11 are used by the BBC Basic ROM and the Electron's keyboard scanning system. Pages 2 and 3 are allocated to the first cartridge slot on the Plus 1 expansion unit, pages 0 and 1 to the second or rearmost slot.

The ElkSD-Plus 1 adds 16KB of RAM to the Electron, present in page 1 or 3, depending on which slot the cartridge is inserted into. This memory is available for use and can be accessed by user programs or those games which support sideways RAM.

The MMFS software is stored in page 0 or 2. The ElkSD-Plus 1 implements a 'split' page for MMFS; most of the 16K page is ROM (physically a flash chip) but the upper 2.5K is RAM that is used for MMFS workspace. This offers better compatibility with games as none of the Electron's main memory is used for workspace. See the next section for more information.

To summarise, with the ElkSD-Plus 1 fitted the Electron's sideways page layout looks like this:

15. Not used
14. Not used
13. Not used
12. Not used
11-08. BBC Basic & Electron Keyboard
07. Not used
06. Not used
05. Not used
04. Not used
03. 16K RAM (first cartridge slot)
02. MMFS (first cartridge slot)
01. 16K RAM (second cartridge slot)
00. MMFS (second cartridge slot)

5. Compatibility & Memory

As mentioned previously, MMFS resides in a sideways page that is split between ROM and 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 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, 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 ElkSD-Plus 1 implements a 'split page' that permits both ROM and RAM in the same page. The ROM section contains the MMFS filesystem, while the RAM section is used by MMFS to store variables. This way a fully functioning copy of MMFS that uses no system RAM can be present while only using up one of the cartridge slot's two 16K pages.

6. Revision 2.x Improvements

Revision 2.x of the ElkSD-Plus1 brings a number of changes from the original 1.x cartridges. Foremost is the new hardware SPI controller. Rev 1.x boards used a 'bit bang' I/O system to communicate with SD card, which employed the Electron's 6502 CPU to handle moving data to the card.

This has been removed on the 2.x board and replaced with a hardware SPI controller inside the main CPLD logic chip. Much of the work required to handle SD card transfers is now done in hardware, resulting in a performance boost of up to 5x for reads and 4x for writes.

For space reasons the extra logic inside the CPLD has necessitated removal of the MMFS selector and flash recovery systems found 1.x boards. The jumper used to configure these capabilities is not present on 2.x boards.

7. Updating MMFS

As and when major MMFS updates are released an updater utility will be made available on the ElkSD-Plus 1 page at www.ramtop-retro.uk

Please make sure to download the ElkSD-Plus 1 Revision 2.x update utility, as 1.x and 2.x boards make use of incompatible versions of MMFS. Running the wrong updater may cause the cartridge to stop working and require service.

If you are unsure which version you have, check for the presence of a jumper on the side of the cartridge. 1.x boards have a jumper, 2.x ones don't.

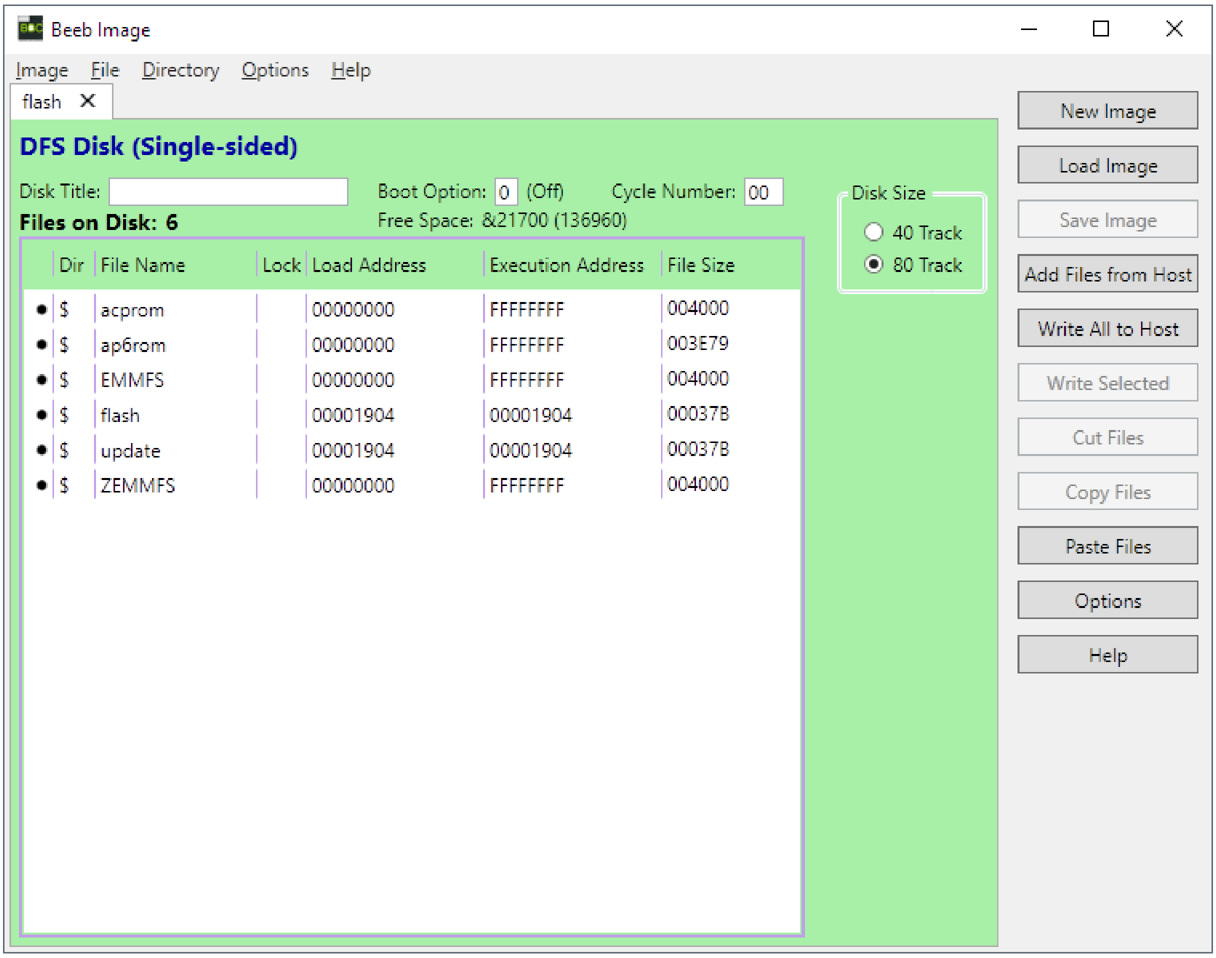
Also, at start-up 1.x cartridges will show 'ESDP1 16K' on the screen, while 2.x ones will show 'SPI 16K'.

8. Adding files to a .SSD Disk Image

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.



DFS Disk (Single-sided)

Disk Title:

Boot Option: 0 (Off)

Cycle Number: 00

Files on Disk: 6

Free Space: &21700 (136960)

Disk Size

○ 40 Track

● 80 Track

	Dir	File Name	Lock	Load Address	Execution Address	File Size
●	$	acprom		00000000	FFFFFFFF	004000
●	$	ap6rom		00000000	FFFFFFFF	003E79
●	$	EMMFS		00000000	FFFFFFFF	004000
●	$	flash		00001904	00001904	00037B
●	$	update		00001904	00001904	00037B
●	$	ZEMMFS		00000000	FFFFFFFF	004000

9. Adding SSD images to 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

10. MMFS Command Reference

See https://github.com/hoglet67/MMFS/wiki/Command-Reference

11. Acknowledgements

The ElkSD64 uses the MMFS software, originally developed by Martin Mather and maintained by David Banks. Many thanks to these gentlemen, without their work this device would not have been possible.