



Loader for Atari 8-bit Computers

User's Manual

Software and documentation ©2022 by Jonathan Halliday

Draft: 310722

Acknowledgements

Without the support of many members of the Atari community (whether in the form of technical assistance, testing, development tools, hardware, material contributions, or sheer moral support), none of what has been accomplished here would be possible. Sincerest thanks to you all, and special thanks to the following (in no particular order):

Hias, Phaeron, Marius, Mr Fish, Kuba, Candle, Lotharek, Mytek, Peter Dell, Dan Baverstock, Drac030, Eric Bacher, Trub, Sergio Larrondo, The Montezuma, tf_hh, Stephen Anderson, Jay Jordon, Albert Yarusso, Jonathan Goring, and Deborah.

Lastly, sincere thanks must be extended to Steve Owens for proof-reading and suggesting technical improvements to this documentation.

Contents

1 Introduction	1
2 The Loader	3
2.1 Starting the Loader.....	3
2.2 Selecting and Opening Items.....	4
2.3 Movement, Selection and Shortcuts.....	4
2.4 Function Modifiers	6
2.5 Joystick Control	6
2.6 Context Menus	7
2.7 The Loader Menus.....	8
2.8 Device Menu	8
2.9 Partition Menu	9
2.9.1 Rename a FAT Partition	9
2.9.2 Get Partition Info	10
2.10 Launcher Menu	10
2.10.1 FAT View	10
2.10.2 File Types	11
2.10.2.1 BASIC files (*.BAS)	12
2.10.2.2 Executable files (*.COM, *.EXE, *.XEX, *.OBJ, *.OBX).....	12
2.10.2.3 Atari Disk images (*.ATR)	12
2.10.2.4 PC Disk Images (*.VHD, *.ISO, *.IMG).....	12
2.10.2.5 Map files (*.MAP).....	13
2.10.2.6 Cartridge files (*.ROM, *.CAR)	13
2.10.2.7 PDM/COVOX audio files (*.PDM, *.PDS, *.COV, *.COS)	14
2.10.2.8 Shortcut files (*.SHC, *.SHn)	15
2.10.3 Directory Sort Order	15
2.10.4 Directory Tree Navigation	16
2.10.5 Handling Large Directories	16
2.10.6 Searching for Files.....	17
2.10.7 APT Partition View	18
2.11 Mounts Menu.....	19
2.12 Tools Menu.....	21
2.13 Options Menu.....	22
2.13.1 Loader Settings	23

2.13.2 Info Menu	24
2.13.3 Exit Menu.....	24
2.14 Advanced Features.....	25
2.14.1 Favourites and Keyboard Shortcuts	25
2.14.2 History.....	26
2.14.3 Autorun.....	27
2.15 File Management	28
2.15.1 Selecting Items	28
2.15.2 Getting File/Folder Information	29
2.15.3 Deleting Items.....	29
2.15.4 Renaming an Item.....	29
2.15.5 Cut, Copy and Paste.....	30
2.15.6 Creating Folders.....	30
2.15.7 Creating Disk Images	31
2.15.8 Changing Item Properties	31
2.15.9 Image and Partition Mounting	32
2.15.10 Map Files and Disk Flipping	32
2.15.11 BASIC/SDX Enable Metatags.....	33
2.15.12 CF/SD Card Hot-Swapping	34
2.16 Error Messages.....	35
3 The Command Processor	37
3.1 Command Prompt	37
3.2 Drives.....	37
3.3 Filenames	37
3.4 Long Filename (LFN) Aliases	37
3.5 Wildcards.....	38
3.6 File Attributes.....	38
3.7 Time/Date Stamps.....	38
3.8 Directories	38
3.9 Pathnames.....	39
3.10 Maximum Path and Command Length	39
3.11 Other Device Names	39
3.12 Default Drive and Directory	39
3.13 Volume Names	40
3.14 Volume Serial Numbers.....	40

3.15 Media Compatibility	40
3.16 Commands.....	41
3.16.1 ATR.....	41
3.16.2 BASIC.....	42
3.16.3 CAR.....	42
3.16.4 CHDIR	42
3.16.5 CHTD	43
3.16.6 CHVOL	43
3.16.7 CLS.....	44
3.16.8 COLD	44
3.16.9 COPY	44
3.16.10 DATE	45
3.16.11 DELTREE	45
3.16.12 DIR	45
3.16.13 ERASE	47
3.16.14 EXIT	47
3.16.15 HELP	48
3.16.16 INFO	48
3.16.17 LOCK.....	49
3.16.18 MEM	49
3.16.19 MEMSAV	49
3.16.20 MKDIR	50
3.16.21 RENAME.....	50
3.16.22 RENDIR.....	51
3.16.23 RMDIR.....	51
3.16.24 RUN.....	52
3.16.25 SPARTA	52
3.16.26 UNLOCK	53
3.16.27 VER.....	53
3.16.28 VOL.....	54
4 Programming with FATFMS	55
4.1 Accessing DOS facilities from BASIC.....	55
4.1.1 Open File.....	55
4.1.1.1 Accessing the Raw Directory	56
4.1.1.2 Attribute Scan Mode	56

4.1.2 Rename File(s)	57
4.1.3 Erase File(s)	57
4.1.4 Protect File(s)	57
4.1.5 Unprotect File(s)	58
4.1.6 Set File Position (Point)	58
4.1.7 Get File Position (Note)	59
4.1.8 Get File Length	59
4.1.9 Load Binary File	60
4.1.10 Change Default Drive and Directory	60
4.1.11 Create a Directory (MKDIR)	60
4.1.12 Delete a Directory (RMDIR)	61
4.1.13 Change Current Directory (CHDIR)	61
4.1.14 Set Attributes (ATR)	62
4.1.15 Set Time Stamp to Current Date and Time (CHTD)	62
4.1.16 Get Disk Information (VOL)	63
4.1.17 Get Current Directory Path	64
4.2 FATFMS User-Accessible Data Table	67
4.3 Error Codes	69
5 Technical Information	71
5.1 File Systems	71
5.2 DOS	71
5.3 MEM.SAV	71
5.4 Implementation Notes	71
6 Future Development	72
6.1 Loader	72
6.2 FATFMS	72
7 Feedback	73
Appendices	74
A References	74
B Glossary	75
C Table of Figures	82
D Index	83

1 Introduction

Building on the success of SIDE/SIDE2, SIDE3 (from Candle O'Sin/flashjazzcat/Lotharek) provides a raft of features including SD-card-based mass storage, powerful cartridge emulation, 2MB of SRAM, DMA, RTC, 8MB of ROM, and all of the facilities for which the earlier SIDE cartridges were known (powerful loader, SpartaDOS X, Ultimate 1MB integration, etc).

SIDE3 – because of its vastly increased ROM space and on-board SRAM – permits new levels of power and convenience in the 'SIDE3 Loader'. Just some of the features provided by the latest version of the loader:

- Read/write support for up to eight FAT partitions with long filenames
- 64KB FAT directory buffer allowing the browsing of up to 1,024 items per folder
- Fast recursive search facility
- Intuitive context menu system
- CAR/ROM file loading, ATR mounting, XEX loading
- Powerful file management tools (recursive file copier, delete, rename, etc)
- Windows-style cut/copy/paste facilities and keyboard shortcuts
- PDM audio playback
- BASIC file loading
- Built-in FATFMS (an Atari/SpartaDOS FAT workalike with CLI)
- Autorun facility
- Bookmarks/favourites
- Launcher shortcuts and aliases
- Disk image creation
- History (remembers your most recently used XEX, ATR and CAR files)
- Support for SD cards/FAT partitions of up to 128GB
- Time/date stamping of new and modified files (via real-time clock)
- Simultaneous HDD/ATR/CAR emulation
- Bootable, OS-level disk image and HDD support with U1MB when present
- Remember the last accessed partition, folder, or file

In an effort to simplify the comparison of various popular multi-cart solutions, the comparison table below shows the different feature sets offered.

	AVG	Ultimate Cart	Uno Cart	SIDE2	SIDE3
Media Type	SD	SD	SD	CF	SD
CAR files up to 1MB	yes	yes	no (up to 128K)	no	yes
XEX Files	yes	yes	yes	yes	yes
Limited ATR support	yes	no	yes	no	yes
Full ATR support	yes*	no	no	yes**	yes**
ATX support	yes*	no	no	no	no
CAS (tape) support	yes*	no	no	no	no
File search	yes	no	yes	yes	yes
Autorun	yes	no	no	no	yes
Bookmarks	yes	no	no	no	yes
File history	no	no	no	no	yes
Disk image creation	yes	no	no	no	yes
Basic file management	yes	no	no	no	yes
Advanced file management	no	no	no	no	yes
CIO FAT DOS	no	no	no	yes (read-only)	yes (R/W)
PDM audio playback	yes	no	no	yes	yes
Video playback	yes	no	no	yes (one file per card)	no
Software firmware update	yes	no	no	yes	yes
Software hardware update	yes	no	no	no	yes
HDD with SpartaDOS X	yes***	no	no	yes	yes
Simultaneous HDD/ATR/CAR	no	no	no	no	yes**
HDD with U1MB	yes***	no	no	yes	yes
User flash ROM	no	no	no	yes (512K)	yes (8MB)
Real-time clock	no	no	no	yes	yes
Battery-backed settings	no	no	no	yes	yes

(*) with optional SIO cable

(**) with Ultimate 1MB

(***) in SIDE2 emulation mode

2 The Loader

The SIDE3 Loader provides access to FAT partitions, allowing the launching of executable programs and the mounting of disk images and cartridges. The loader can also open the 'APT' on a disk, facilitating the dynamic mounting of hard disk partitions.

2.1 Starting the Loader

There are several ways to start the SIDE3 loader. Once the cartridge is plugged into the Atari, you can:

- Turn on the computer with the two-position cartridge switch in the loader (upper/rear) position
- Turn on the computer with the two-position cartridge switch in the SDX (lower/front) position, and when SpartaDOS X boots, type 'CAR' at the command prompt
- While other software is running, press the cartridge reset button, and then press the Atari's system reset key
- When Ultimate 1MB with the SIDE3 plugin is present, press 'L' at the start-up screen or in the U1MB BIOS set-up menu
- When Ultimate 1MB with SIDE3 plugin is present, configure the system to 'Boot to loader'
- From the U1MB 'Boot drive' menu, select 'Loader'

Assuming you have inserted a FAT16 or FAT32 formatted SD card containing some files and directories into the SIDE3 cartridge, upon starting the loader, you will see something similar to this:



Figure 1: The browser/launcher menu

The loader has eight menus, and by default – if the loader finds a suitable FAT16/32 partition on the card (or there is more than one such partition and the loader is configured to automatically open the last accessed partition) – it will open the third menu (the 'browser/launcher') when it starts. The launcher menu presents an alphabetised list of all the folders and supported files on the card (with folders grouped together at the top of the list). Although the default configuration is to display only files which the loader can actually open, it's possible to force the loader to display all (even hidden) files (see the 'Loader Options' section).

Folders are signified by a 'folder' icon in the left margin, while supported files are identifiable by their three-character file extension, which is always visible at the right hand side of the line, even if the filename is too long to fit on the screen in its entirety (in which case, the start and end of the name will

be displayed, with omitted characters in the middle signified by an ellipsis; such long names will automatically scroll left and right to reveal themselves in full when highlighted with the selection bar).



Figure 2: A scrolling long filename

2.2 Selecting and Opening Items

To move the highlight up and down the list, simply use the UP and DOWN arrows (holding CONTROL at the same time is optional). You can move by a complete 'page' at a time by pressing the UP/DOWN arrow keys while holding down the SHIFT and CONTROL keys simultaneously.

To open the highlighted item, press the RETURN key, or the trigger button if using a joystick.

2.3 Movement, Selection and Shortcuts

You may control the loader using the keyboard or joystick (joystick control must be enabled on the appropriate port in the loader settings in order for the latter to work; see the 'Loader Options' section later in this chapter). Below is a complete list of keyboard/joystick shortcuts.

Action	Keyboard	Joystick	1200XL F-Key
Previous Menu	LEFT ARROW	LEFT	F3
Next Menu	RIGHT ARROW	RIGHT	F4
Previous Item/setting	UP ARROW	UP	F1
Next Item/setting	DOWN ARROW	DOWN	F2
Page Up	SHIFT+CTRL+UP ARROW		SHIFT+F1
Page Down	SHIFT+CTRL+DOWN ARROW		SHIFT+F2
Open	RETURN	Button	CTRL+F3
Open with modifier/mount	CTRL+RETURN		CTRL+F4
Mount image	TAB		
Select item	CTRL+SPACE		CTRL+F1
Assign drive number to image	CTRL+Number (1-9)		

Action	Keyboard	Joystick	1200XL F-Key
Cancel operation, exit search mode, or back up to the parent directory	ESC	LEFT with button	
Delete the previous character of the search phrase	DELETE/BACKSPACE		
Clear the search phrase	CTRL+CLEAR		
Abort search	BREAK		
Select all	CTRL+A		
Add to Autorun	SHIFT+CTRL+A		
Toggle BASIC	CTRL+B		
Copy item(s)	CTRL+C		
Delete item(s)	CTRL+D		
Refresh disk	SHIFT+CTRL+D		
End/bottom of list	CTRL+E		
Add item to Favourites	CTRL+F		
Open Favourites folder	SHIFT+CTRL+F		
Go to containing folder	CTRL+G		
Home (root) Folder	CTRL+H		
Open History folder	SHIFT+CTRL+H		
Get item information	CTRL+I		
New disk image	SHIFT+CTRL+I		
Open context menu	CTRL+M		
Make new folder	SHIFT+CTRL+M		
Rename item	CTRL+N		
Change sort order	CTRL+O		
Show in target folder	SHIFT+CTRL+O		
Parent Folder	CTRL+P or ESC		
Set item properties	SHIFT+CTRL+P		
Restart	CTRL+R		
Open Autorun folder	SHIFT+CTRL+R		
Swap disk image mount points	CTRL+S or cart button		
Boot SpartaDOS X	SHIFT+CTRL+S		
Start/top of list	CTRL+T		
Unselect all	CTRL+U		
Undo mounts	SHIFT+CTRL+U		
Cut item(s)	CTRL+X		
Undo	CTRL+Z		

When the joystick is used to control the loader, input will auto-repeat just as it does when using the keyboard. In addition, a type-ahead buffer ensures that keyboard and joystick input is never lost while the system is busy.

When the 'accelerate' setting is enabled (see the later section on the 'Options' menu), the auto-repeat rate will be reduced after twenty repetitions. This facilitates fast movement through the file list.

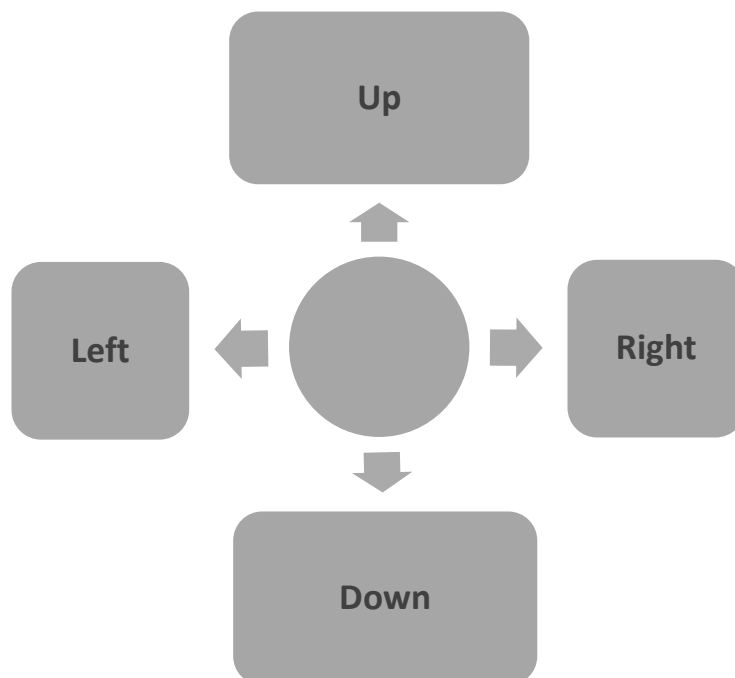
2.4 Function Modifiers

CTRL held with the RETURN key has a special meaning depending on the selected item's type and the context of the operation.

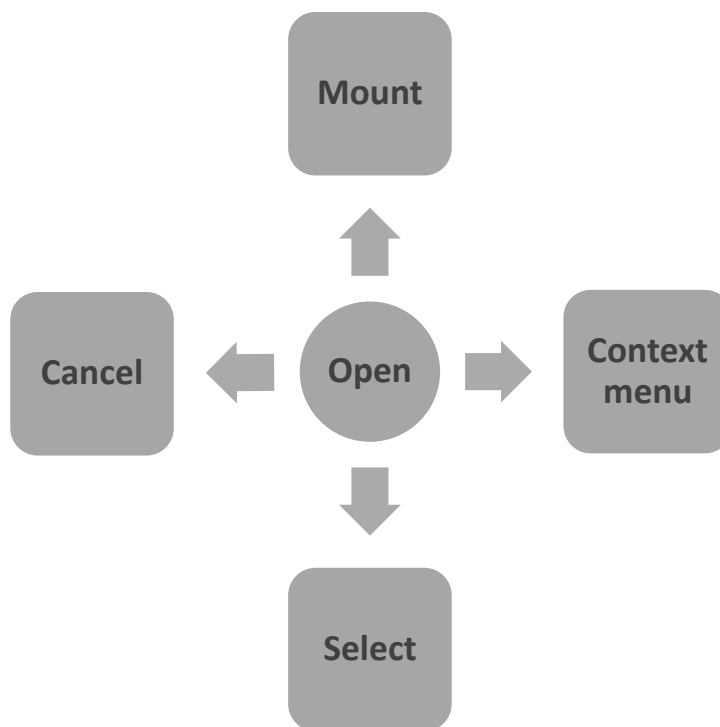
Item type	RETURN	CTRL+RETURN
XEX file in logged folder	Run in logged folder	
XEX file in search results	Run in own folder	Run in logged folder
ROM or CAR file (SIDE3 only)	Mount and reboot	Mount
BASIC program in logged folder	Run in logged folder	
BASIC program in search results	Run in its own folder	Run in current folder
ATR	Mount and reboot	Mount
MAP file	Mount and reboot	Mount

2.5 Joystick Control

Joystick actions with the trigger released are as follows:



Joystick actions with the trigger held down are as follows:



Note that standard trigger actions are carried out when the trigger is released with the joystick in the neutral position. Therefore, to access the actions in the diagram above, press the trigger, move the joystick in the required direction, and release the trigger before returning the joystick to the neutral position.

2.6 Context Menus

The latest version of the SIDE3 Loader includes context menus which are invoked with CTRL+M, or by moving the joystick to the right with the button held down.



Figure 3: Launcher context menu

Once a context menu is open, it may be navigated with the UP/DOWN arrow keys or the joystick, and you can jump forward or back by ten items with SHIFT+CTRL+UP/DOWN. Select the desired item by pressing RETURN or the joystick button. Press ESC or the LEFT/RIGHT arrow keys (or the same axes on the joystick) to close the menu without choosing anything. If the context menu contains more items than will fit on the screen, the menu will automatically scroll up and down, and an indicator in the menu's left margin shows when there are unseen items beyond the top or bottom of the visible list.

2.7 The Loader Menus

The next eight sections will cover each of the loader's eight menus in turn. You can move between menus with the left and right arrow keys (or via the left and right joystick axes). The active menu is denoted by the highlighted icon in the icon bar, and by the title directly beneath that.

2.8 Device Menu

The device menu displays the physical controllers attached to the system (one per line), each followed by the name and model of the inserted SD card, if present.



Figure 4: The Device Pane

Opening the selected device (with the RETURN key, or by opening the context menu and selecting 'Open') causes the loader to build a list of all partitions on the media, and (optionally) to open one of the partitions and display its contents in the Launcher menu. If only one FAT partition exists on the card, the Launcher menu will always immediately open that partition when the device is opened. However, if several partitions exist on the card, a partition will only be automatically opened in the launcher menu if the loader is configured to remember the last accessed partition/folder/file, and said last accessed item exists on the currently inserted media. Otherwise, opening the device will open the Partition menu instead, inviting the user to select a FAT partition (or APT) for browsing.

2.9 Partition Menu

The second menu is the partition menu, which displays a list of all the recognised partitions on the disk (which includes FAT16 partitions, FAT32 partitions, and the APT container, if present). The partition menu takes the name of the inserted media as its title (otherwise – if no media is present – ‘No media’ is displayed as the title).



Figure 5: Partition Menu with context menu open

After navigating to the desired entry with the cursor keys or joystick, pressing RETURN or the joystick button will open the partition in the browser. As with the device menu, pressing CTRL+M or moving the joystick down with the button pressed will open the context menu, which here offers two options in addition to ‘Open’: ‘Rename’ and ‘Get Info’.

2.9.1 Rename a FAT Partition

Selecting ‘Rename’ from the context menu or pressing CTRL+N places the name of the highlighted FAT partition into edit mode so that you can edit the existing name or provide a new one. Pressing RETURN after typing the desired name will cause the partition to be renamed, while pressing ESC will abandon the operation. Note that the APT cannot be renamed since it has no name (it is labelled ‘Untitled’).

2.9.2 Get Partition Info

Selecting 'Get Info' from the context menu or pressing CTRL+I will cause information about the highlighted FAT partition to be displayed. The information includes the FAT partition number, the volume name and serial number, the file system (FAT16 or FAT32), the volume size, free space and the cluster size.



Figure 6: Partition Info

2.10 Launcher Menu

The Launcher menu (symbolised by a 'rocket' icon) takes as its title the volume name and current path of the open FAT partition (or 'APT' when the APT is being browsed). Commonly, this is the menu which is displayed by default when the loader starts.

2.10.1 FAT View



Figure 7: A FAT Partition open in the Launcher

Most of the loader's functions are carried out in the FAT view in the launcher; this is where you run XEX files, mount disk and cartridge images, browse directories, search the volume, and manage your files.

Long filenames are displayed and names too long to fit entirely on the screen are abbreviated with an ellipsis ('...') in the middle of the name. When you highlight such an entry, the ellipsis disappears and the entire name instead scrolls left to right automatically in order to fully reveal itself. The three-character extension of a recognised file type never moves, however, and always remains in a fixed position at the right-hand side of the line.

Folders are denoted by a 'folder' icon in the left margin, and any currently mounted disk images (ATRs, ISOs, VHDs, etc) will be preceded by their current mount point (i.e., their logical drive number).

If 'Show size' is enabled in the 'Options' menu (see the section on 'Options'), files will also have their size displayed at the right-hand side of the screen.

Any selected (marked/tagged) files or folders (see the 'Advanced Features' section later in this chapter) will be denoted by a check mark at the extreme right of the screen (and the rest of the information will shift left to accommodate it).

Lastly, if more than sixteen entries exist in the current folder, a scroll bar handle will appear at the extreme right of the screen. This scroll bar handle indicates the position and size of the current 'page' of sixteen entries in relation to the entire list.

2.10.2 File Types

The launcher displays the following file types:

Extension	File Type
XEX	Binary executable
EXE	Binary executable
COM	Binary executable
OBJ	Binary executable
OBX	Binary executable
ATR	Atari disk image
VHD	Windows virtual hard disk image
ISO	Disk image
IMG	Disk image
BAS	Tokenised BASIC program
MAP	Script describing multiple ATR mount points
ROM	Raw cartridge ROM image
CAR	Cartridge ROM image with header describing banking scheme/geometry
PDM	Pulse Density Modulated audio (FujiConvert) files

Extension	File Type
PDS	Stereo PDM files
COV	Covox files
COS	Stereo Covox file
SHC	Shortcut file

The file types are described in detail below.

2.10.2.1 BASIC files (*.BAS)

Tokenised BASIC programs (typically run under Altirra BASIC or Atari BASIC) may be launched by the loader by means of the built-in FAT file system handler. If the file system is disabled, it will nevertheless be installed when running a BASIC program, since it is via the FMS that the BASIC program is loaded. Once the BASIC program is running, it has full read/write access to the FAT partition from which it was launched. The CIO FMS handler is reset-proof, and calling DOS from a BASIC program launched in this manner will transfer control back to the loader. See Chapter 3 ('The Command Processor') for information on running BASIC programs from DOS.

2.10.2.2 Executable files (*.COM, *.EXE, *.XEX, *.OBJ, *.OBX)

Executable files are loaded by code residing at 0x0700 which does not require the file system handler (FATFMS) to be present. However, enabling the file system handler provides launched programs with full CIO read/write file access to the FAT partition. One application for this is launching UFLASH.XEX from the XEX loader, flashing ROM images to U1MB or even dumping them back to disk.

When the file system handler is not present (i.e., disabled in the Options menu; see the 'Loader Options' section), MEMLO resides at approximately 0x0B00 (the loader code being some 1KB long). When the file system handler is enabled, MEMLO sits at 0x2000, although XEX files may still overwrite anything beyond the end of the loader (0x0AFF) without problems during the launching process. If an application is designed to work with DOS (i.e., loads beyond 0x2000 and implements standard CIO calls), it should load and work with the FATFMS.

2.10.2.3 Atari Disk images (*.ATR)

Atari disk images (ATRs) can be mounted a) if 'ATR support' is enabled in the Options menu, or b) if U1MB is present and the PBI HDD is enabled. After mounting an ATR, the loader will usually restart the operating system and boot the system from drive 1 (the restart can be overridden by using CTRL+RETURN instead of RETURN).

2.10.2.4 PC Disk Images (*.VHD, *.ISO, *.IMG)

With U1MB and a corresponding SIDE3 PBI BIOS update, it's possible to mount VHD, ISO and IMG PC/Mac compatible disk images (the loader's 'soft OS' ATR functionality does not support this kind of disk image). This functionality is provided to support future versions of FATFMS.

2.10.2.5 Map files (*.MAP)

Map files describe the mount points for ATRs. Map files are plain DOS/Windows/Unix text files containing lists of drive specifiers and ATR filenames. See the section on Image and Partition Mounting for more information.

2.10.2.6 Cartridge files (*.ROM, *.CAR)

The SIDE3 cartridge is capable of mounting cartridges of many different types in CAR and ROM formats. CAR files are preferred since they include a header with a description of the banking scheme (if any) used by the cartridge, whether it is a left or right slot cartridge, whether it is an 8K or 16K cartridge, etc. CAR files – if of a supported type – will automatically mount without any further input from the user. ROM files lack the header information, so unless the loader can guess the cartridge type from the size of the file or other characteristics, it will present you with a list of compatible cartridge types from which you should choose whatever is appropriate for that cartridge image.

Cartridge images can be made ‘persistent’ (i.e., only being unmounted when you explicitly unmount the image) or can be forced to unmount automatically every time you restart the loader. See the ‘Unmount carts’ setting in the ‘Options’ menu. By default, a mounted cartridge automatically unmounts when you re-enter the loader.

By default (when cartridge images are launched with the RETURN key), a reboot is immediately performed. If you wish to mount a cartridge image without an immediate restart, simply mount the image with CTRL+RETURN. You can then perform other tasks (such as mounting ATRs, etc) or reboot into SpartaDOS X with SHIFT+CTRL+S.

2.10.2.7 PDM/COVOX audio files (*.PDM, *.PDS, *.COV, *.COS)

The SIDE3 Loader can play mono and stereo Pulse Density Modulated (PDM) and Covox audio files created by 'FujiConvert' (<https://lybrown.github.io/fujiconvert/>). Select 'IDE Player (flashjazzcat)' as the output format in the FujiConvert tool:

FujiConvert

Settings

Resampling Window: 1024 ▾

Dither: ☐

Auto-gain: ☒

Gain: 1

CPU Speed: 1

Offset: 0

Duration: -1

Title:

Artist:

Max Size: 128M ▾

Generate .WAV Preview: ☐

PDM Settings:

Preset: 16 16 0 ▾

DC Offset: -7 ▾

Coarse Levels: 16 ▾

Fine Levels: 14 ▾

Bump: 0 ▾

Non-Linear Pulse: 2/4

Linear Pulse: 3/5

Output Media:

☐ RAM XEX

☐ Emulator Stream XEX

☒ IDE player (flashjazzcat)

☐ The!Cart

☐ Atarimax

☐ MegaMax

☐ MegaCart

☐ XEGS

☐ SIC!

Restore Defaults

Playback Method:

☒ PDM

☐ PCM

☐ PWM

☐ Covox at \$D600

Channels:

☒ Mono

☐ Stereo

Region:

☒ PAL

☐ NTSC

Frequency:

☐ 47kHz

☐ 44kHz

☐ 33kHz

☒ 31kHz

☐ 22kHz

☐ 15kHz

☐ 8kHz

Select Input Media

Choose File yt1s.com - ...s Theme.mp3 Reconvert

Constrained Settings

```

title:
artist:
player_name: undefined
resampling_window: 1024
autogain: true
bump: 0
cart_type: raw
channels: mono
coarselevels: 16
dc: -7
dither: false
duration: -1
finelevels: 14
freq: 44270.27027027027
frequency: 44kHz
gain: 1
linpulse: 3/5
maxsize: 128M
media: ide
method: pdm
nonlinpulse: 2/4
offset: 0
period: 37
preset: 16 16 0
region: pal
speed: 1

```

Read and Decode

=====

Mix and Resample (Auto-gain=1.521)

=====

Convert

=====

Zip

=====

Download

Uncompressed size: 6333221 bytes

Link: yt1s.com - Ronnie Hazlehurst His Orchestra The Two Ronnies Theme pdm14 mono 44270Hz ide pal.zip

Preview: Play Stop

By Xuel. 2019. MIT License. Version 0.3.3 [Github](#)

Needless to say, Covox playback is only possible on a Covox-equipped machine. You should use the following naming convention on files created with FujiConvert:

Extension	File Type
PDM	44KHz mono PDM
PDS	22KHz stereo PDM
COV	44KHz mono COVOX
COS	22KHz stereo COVOX

You may specify the base address of the Covox hardware with the 'Covox base' setting in the 'Options' menu. PokeyMAX users with a Covox-enabled core (and Covox enabled in their PokeyMAX configuration) will find the 'Covox base' setting auto-populated and greyed out (so there is no need to manually set the Covox base address).

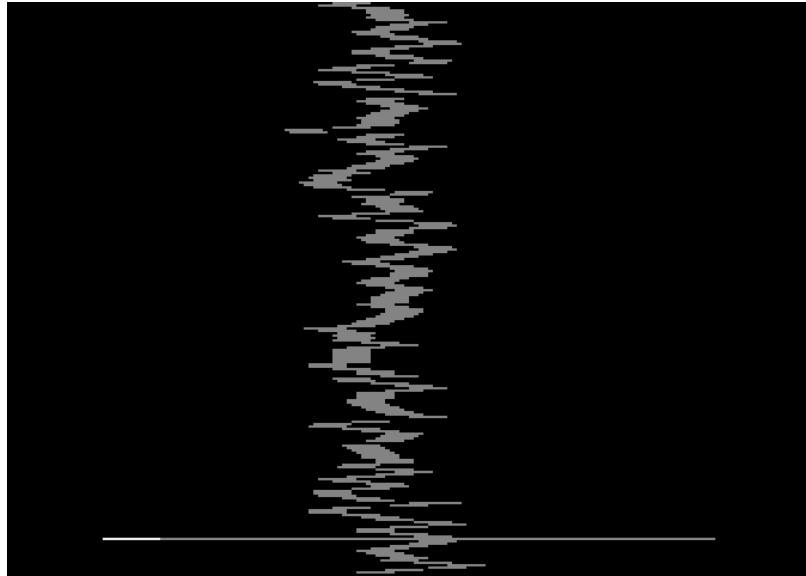


Figure 8: The PDM Player

In the SIDE3 PDM player, you can press the left and right arrow keys during playback to rewind and fast forward, and tap the space bar to pause and resume playback. The ESC key abandons playback and takes you back to the file browser.

2.10.2.8 Shortcut files (*.SHC, *.SHn)

Shortcut files are aliases which point to other files or folders (see 'Advanced Features' section information on how to create shortcuts and optionally register them to hotkeys). They usually share the name of the target file, but with a '.SHC' or '.SHn' suffix (where 'n' is the number of a hotkey shortcut slot, of which there are ten). Shortcuts may exist anywhere in the directory structure, but are most commonly created in three special (hidden) folders: 'autorun', 'favourites' and 'history'. Shortcuts may be freely copied from one special folder to another, but hotkey shortcuts will only be registered to their corresponding keyboard shortcut (SHIFT+CTRL and the shortcut number) when they reside in the 'favourites' folder.

2.10.3 Directory Sort Order

Although FAT directories are by default displayed in alphabetical order (by name) with folders grouped before files, the sort order can be temporarily changed by selecting 'Sort order' from the context menu, or pressing CTRL+O.



Figure 9: Changing the sort order

The file list will be re-sorted immediately using the chosen method. Note that the change is not permanent and the next time a directory is read, the sort order will revert to the method dictated by the 'Sort order' setting in the 'Options' menu. You should set the desired sort order there to make a change to the default sort order permanent (see the 'Loader Options' section).

2.10.4 Directory Tree Navigation

Subdirectory names in the launcher menu are (by default) grouped together at the beginning of the file list and are prefixed with 'folder' icon. To open a subdirectory, simply highlight it and press RETURN or the joystick button. Once inside of a subdirectory (i.e., not in the root directory), two additional entries will appear right at the top of the file list: the parent folder entry (a crooked left-pointing arrow) and the root folder entry (a straight upward-pointing arrow). Opening the parent folder will take you one level back up the folder tree and opening the root folder entry will take you back to the root folder. Note that when you enter a subdirectory and then back up out of it again, you will return to the previous position in the parent folder.

You may reach the parent of the logged folder by pressing CTRL+P (as an alternative to ESC), and the home (root) folder by pressing CTRL+H. The 'home' shortcut may be used even when in search mode; it will cancel the search and return to the top of the root directory.

2.10.5 Handling Large Directories

The SIDE3 loader will read up to 1,024 filenames (spanning 64 'pages') or 64KB of filename data (whichever limit is reached first) in a single directory. It is hoped this will be enough for most situations, but if not, it is suggested that such huge directories (which take longer to read) are broken up into smaller folders. Note that filenames which do not fit into RAM will simply be discarded and the partial directory displayed (remember also that displaying 'all' and/or 'Hidden' files can significantly increase the number of filenames loaded into memory).

When reading directories containing more than 256 filenames, the loader will update the total filename count during the reading process for every additional 256 filenames in order to provide an indication of progress.

2.10.6 Searching for Files

Rather than paging through a long list of filenames, the loader allows rapid searching for a partial filename match simply by typing characters. The search is 'recursive' (i.e., it scans the current directory and all subordinate directories), and returns results from the current directory first (before processing any other folders). It's possible to move through the file list and perform certain actions while the list of results is growing, although certain functions which require disk IO will halt the search if carried out. Editing the search criteria will always re-start the search, and you can halt the search at any time with the BREAK key (leaving the results on the screen), or abandon it entirely and return to the current folder via the ESC key. When all characters have been deleted from the search phrase (either with DELETE/BACKSPACE or CTRL+CLEAR), an extra press of the DEL or ESC key is required to actually leave search mode.

Once the search has completed, you may perform almost all tasks on the list of results which you can perform on files in a standard directory (with the exception of operations – such as creating a new directory – which rely on being in a 'current' folder). Therefore, if you see what you're looking for while the search is ongoing, you can immediately cursor to it and open the item without waiting for the search to complete. Likewise, you can select files or groups of files for copying and deletion (even if they are dispersed all over the directory tree), display their properties, etc. You could, for example, isolate a group of files via the search facility, select them, copy them, and then paste them all to a new folder.

As an example: to locate all files and folders whose names include the word 'TUR', simply type the phrase and the search will begin automatically. To narrow or broaden the search, just type additional characters or shorten the string with the DELETE/BACKSPACE key.



Figure 10: Searching for 'TUR'

Occasionally, one may wish to enter characters into the search string which are normally used as navigation keystrokes (for example, the '-' or '_' characters). To enter such characters in the search string, simply press CTRL+ESC first.

To completely clear the search string without actually leaving search mode, press CTRL+CLEAR or SHIFT+CLEAR.

In the list of search results, you can open the containing folder of any entry (file or folder) by highlighting it and pressing CTRL+G (or choosing 'Goto location' from the context menu).

Note that very ambiguous search criteria (such as a search for 'XEX' which would match every XEX file on the volume) may result in a very large number of search results which might quickly exceed the capacity of the filename buffer (1,024 names).

2.10.7 APT Partition View

The launcher's other 'view' is 'APT Partition View', and you'll see this when opening the APT container via the partition menu. After opening the APT, you'll see a list of all Atari partitions complete with their current drive assignments, partition names (where present), partition IDs, partition types, and sizes.

Mounting and launching work much as they do when XEX and ATR files are displayed in FAT View, although there are no facilities for copying, renaming, deletion, etc.

In the partition type column, you'll see 'DOS' or 'Ext' since these are the only two APT partition types catalogued by the loader. 'DOS' denotes a standard partition, while 'Ext' denotes an 'Extended' APT partition: namely one which is mapped directly to an MBR FAT partition located outside of the APT area. External entries (which are described in the APT partitioning manual) may be created via the APT FDISK partitioning tool, and they provide an excellent method of sharing data between PC operating systems (such as Windows and Linux) which recognise and mount MBR partitions, and Atari disk operating systems (such as SpartaDOS X) which can read FAT partitions (although the FATFMS 'DOS' present in the loader provides full read/write access to FAT16 and FAT32 partitions independently of whether they are mounted as 'external' APT partitions). External entries also provide an essential means of assigning drive numbers to MBR FAT partitions at the partitioning stage (since the Atari – unlike a PC – needs to store persistent drive assignments on the disk).



Figure 11: APT in launcher with context menu open

At the extreme right, the logical partition size is shown. Note that this size does not necessarily represent the physical size of the partition on disk. Rather, it is the addressable size of the partition. Although a 65,535-sector partition occupies 32MB on disk, if the logical sector size of the partition is 256 bytes, the size will be shown as 16MB.

2.11 Mounts Menu

The Mounts menu displays mounted cartridge media and a list of logical drive assignments (for drives 1 through 15).

Drives may have one of three states: SIO, HDD, or IMG. 'SIO' simply means the drive number refers to a device on the SIO chain (regardless of whether any such device actually exists on that drive number). 'HDD' means the drive number refers to an APT hard disk partition, and 'IMG' means the drive is mapped to a mounted disk image (ATR, ISO, VHD or IMG).

APT entries (partitions) have their partition ID and (optionally) partition name displayed next to them. The partition ID is the same one displayed alongside each partition entry in the launcher menu. ATR entries, meanwhile, have their host partition ID displayed alongside, followed by the name of the mounted ATR (if said filename has been picked up during directory scans). The host partition ID corresponds to the ID displayed on the Partition menu next to the FAT which contains the disk image file. Therefore, several ATRs may share the same host partition ID.



Figure 12: The Mounts menu showing multiple mounted media

ATR and HDD names are only displayed if they happen to have been catalogued on the launcher menu (by logging the ATR's containing folder, opening the APT, or running a recursive search which happens to match the mounted ATR). This opportunistic approach to populating names in the drives menu is preferred to recursively scanning every folder in every MBR partition every time the loader is launched with ATRs already mounted.

It's possible to reveal the image mounted on a given drive slot in its original location in the FAT directory structure or APT. Pressing CTRL+G (or selecting 'Goto location' from the context menu) will display the highlighted disk image or APT partition in its containing folder/APT.



Figure 13: Performing a reverse lookup on a disk image

In the case of an image, finding the containing folder may involve a scan of the entire folder tree, although if the corresponding ATR exists in the currently logged directory or search results, location of the file will be instantaneous.

This feature can be rather useful if the name of the ATR or partition has not populated the slot of interest on the Drives menu. In that case, pressing CTRL+G on the entry will take you directly to the image mounted on that drive number, simultaneously populating the name field on the Mounts menu in the process.

Pressing RETURN on an entry will unmount the media.

2.12 Tools Menu

The Tools menu has three items: 'Unmount images', 'ATR swap', and 'Disk refresh'.

'Unmount images' simply re-reads the partition table from disk, unmounting all disk images and any cartridge, and restoring the partition map to the state it was in when the computer was first booted (or the last time changes were written to the partition table by the FDISK tool, or the last time the partition table was refreshed). You may access 'Unmount images' from any of the loader menus by pressing SHIFT+CTRL+U.



Figure 14: Tools menu

The 'Swap ATRs' option rotates (swaps) all mounted disk images by one place. For example, if ATRs are mounted on drives 1, 2 and 3, rotating or swapping the ATRs will move drive 1's ATR to drive 3, drive 2's ATR to drive 1, and drive 3's ATR to drive 2). This is exactly equivalent to rotating ATRs by pressing the ATR swap button on the SIDE3 cartridge. You can rotate ATRs in this manner anywhere in the loader by pressing CTRL+S.

Selecting 'Disk refresh' (you can also press SHIFT+CTRL+D anywhere in the loader) will issue a card reset, and refresh the partition table and FAT/APT catalogues.

2.13 Options Menu

The Options menu provides a large number of user-configurable settings.



Figure 15: Changing the 'Sort by' setting

All of the loader's default settings can be changed here, including the current system date and time (used for time and date stamping new and updated files).



Figure 16: The second page of the Options menu

All settings 'take' as soon as they are changed, although changes to editable settings with drop-down lists may be cancelled with the ESC key (RETURN being used to select and keep changes). Changes to settings are immediately saved to the non-volatile (battery-backed) RAM on the SIDE3 cartridge, so that they will survive a power cycle.

2.13.1 Loader Settings

Below is a complete list of all available loader settings.

Option	Purpose	Settings	Default
BASIC	Toggle the presence of internal BASIC (and the external SIDE3 cartridge ROM where available) when launching XEX or ATR files	Enabled, Disabled	Disabled
FAT FMS	Toggle the presence of the 'FATFMS' CIO file system handler (DOS)	Enabled, Disabled	Disabled
Show size	Toggle display of XEX/ATR and logical volume sizes	Enabled, Disabled	Enabled
Show all files	Toggle display of all files when reading directories (not just those understood by the loader)	Enabled, Disabled	Disabled
Show hidden/sys	Toggle the display of hidden and system files	Enabled, Disabled	Disabled
Alias extenders	Toggle the display of alias extenders (*.SHC)	Enabled, Disabled	Disabled
Modal search	Toggle the display of shortcuts next to filenames (makes search modal)	Enabled, Disabled	Disabled
Sort by	Specify the sort order of files/partitions	None, Name, Type, Size, Date/Time	Name
Reverse sort	Toggle reversal of chosen sort method	Enabled, Disabled	Disabled
Joystick	Change the port for joystick control or disable it entirely	Port 1, Port 2, Disabled	Port 1
Colour	Select one of sixteen colour schemes	0-15	0
Sound	Enable or disable the key click sound	Enabled, Disabled	Enabled
Screensaver	Toggle the five-minute screen timeout	Enabled, Disabled	Enabled
Accelerate	Toggle cursor auto-repeat acceleration	Enabled, Disabled	Enabled
History	Toggle recording of execution history	Enabled, Disabled	Disabled
Remember	Specify which 'last accessed' item to remember when the loader starts	Partition, Folder, Item	Partition
Unmount carts	Specify if or when the loader should automatically unmount any cartridges when other media is opened or mounted	Always, Ask, Never	Ask

Option	Purpose	Settings	Default
Date	Set the system date		
Time	Set the system time		
ATR support	Toggle support for disk image mounting	Enabled, Disabled	Enabled
Autorun	Toggle the autorun facility	Enabled, Disabled	Disabled
Extended RAM*	Toggle 512KB RAM expansion	Enabled, Disabled	Disabled
PBI BIOS*	Toggle PBI BIOS	Enabled, Disabled	Disabled
Hard emulation	Toggle 'hard' cartridge emulation	Enabled, Disabled	Disabled
Covox base	Set the base address of Covox hardware	None, 0xD100, 0xD280, 0xD500, 0xD600, 0xD700	None
DOS MEM.SAV	Toggle FATFMS's 'MEM.SAV' facility	Enabled, Disabled	Disabled
Optimise FAT	Toggle FAT allocation optimisation	Enabled, Disabled	Enabled
SpartaDOS ID	Toggle FATFMS SpartaDOS ID at \$700	Enabled, Disabled	Disabled

*Requires hardware rev. 3.1 and JED update

2.13.2 Info Menu

The Info menu displays the loader version number, copyright notice, and various items of information about the loader and SIDE3 hardware, including the hardware and JED revision and the amount of SRAM on the board (between 2 and 8MB).

2.13.3 Exit Menu

The Exit menu contains the 'Reboot' and 'Boot SpartaDOS X' items. Both of these functions may be invoked from anywhere in the loader by pressing their shortcut keys.

2.14 Advanced Features

The SIDE3 loader offers several facilities which make automation and file organisation easier. You can autorun items, bookmark your most frequently used files, and maintain a history of the most recently accessed items. You can also set up keyboard shortcuts for up to ten items.

The loader maintains up to three ‘special’ hidden folders (‘favourites’, ‘history’ and ‘autorun’), and the contents of these special folders may be manipulated in just the same way as any other items (i.e., they may be renamed, deleted, copied, etc) via the file management facilities described later in this section.

2.14.1 Favourites and Keyboard Shortcuts

The ‘Favourites’ feature allows the creation of shortcuts to commonly accessed files and folders. To make a favourites shortcut for the highlighted item, select ‘Add favourite’ on the context menu, or press CTRL+F.



Figure 17: ‘Add favourite’ dialog

A list of options will then appear, allowing a standard ‘favourites’ shortcut to be created, or one of ten ‘hotkey’ shortcuts which can later be invoked with SHIFT+CTRL and the corresponding number key. Whichever option you select, if a shortcut with the same name already exists in the favourites folder, you will be asked whether you want to replace it with the new one, and you can back out of the operation with the ESC key at any point. If the hidden favourites folder does not yet exist when the new shortcut is created, the folder will be created first.

Depending on what kind of shortcut you select, shortcut files are named after the target file with either a ‘.SHC’ extension (following the original extension), or ‘.SHn’ where ‘n’ is the hotkey number. You can manually convert a non-hotkey shortcut (*.SHC) to a hotkey shortcut by manually changing the file extension (select ‘rename’ with the shortcut file highlighted).



Figure 18: Editing a shortcut extender

To access the favourites folder, select 'Open favourites' from the context menu, or press SHIFT+CTRL+F. If the favourites folder doesn't exist, an error message will be displayed. Otherwise, you'll be presented with the folder's contents. Shortcut files are denoted by a curved arrow in the left margin, and by default their 'SHC' file extensions are hidden. The rest of a shortcut's name is exactly the same as the item to which the shortcut refers. Shortcuts may point to any item type supported by the loader, and pressing RETURN or selecting 'Open' will run or mount the item pointed to by the shortcut. If the shortcut points to a folder, opening the shortcut will open the target folder in the launcher.

2.14.2 History

The loader can optionally maintain a 'history' of the most recently opened items. The history feature (which is disabled by default) must first be enabled in the loader's options menu. Once enabled, the history feature will create a shortcut file to every opened or mounted item and place it in the hidden 'history' folder in the root directory of the open FAT volume. The size of the history folder is limited to 100 entries, and older entries will be deleted automatically as new history shortcuts are created if this limit is exceeded. If the history folder does not exist when the first history shortcut is created, the folder will first be created.

To open the history folder, select 'Open History' in the context menu, or press SHIFT+CTRL+H. If the folder doesn't exist, an error message will be displayed. Otherwise, you'll be presented with the contents of the history folder.



Figure 19: The History folder

Shortcuts in the history folder are by default arranged in reverse chronological order, so that the most recently added entries appear at the top of the list. You can change this by selecting 'Sort order' in the context menu (although the history folder will always revert to reverse chronological sort order when re-opened). Shortcuts are opened exactly as per those in the favourites folder.

Since duplicate entries will be automatically overwritten, opening an item in the history folder (from anywhere in the history) will cause a new shortcut to the target item to appear at the top of the list.

2.14.3 Autorun

The loader is capable of 'autorunning' items (files and folders) by placing shortcuts to the items in the hidden 'autorun' folder.



Figure 20: Setting up autorun on an executable

To create an autorun shortcut to the highlighted item, choose 'Autorun' from the context menu, or press SHIFT+CTRL+A and respond 'OK' to the resulting prompt. A shortcut to the target item will be created in the hidden 'autorun' folder (and the folder will be silently created first if it does not exist).

To complete the autorun process, one must enable 'Autorun' in the loader's options menu. Thereafter, the first time the loader is started after a cold power-up, it will attempt to launch the items pointed to by shortcuts in the autorun folder. In the case of executable files, disk images and cartridge ROMs, the loader's user interface will not appear at all; the system will boot directly into the chosen media. To get back to the loader itself, invoke it by the usual methods and the user interface will appear (you can then disable the autorun feature again if you want the loader to start as normal after the machine is powered on).

Multiple autorun shortcuts (of different file types) can be created, and these will be processed in order. Conflicting autorun entries (i.e., files of the same type) will be deleted if a new shortcut to the same file type is created (you may autorun an ATR and CAR file at the same time, for example, but not two CAR files at once). You may create an autorun entry for an existing shortcut in the favourites or history folder (a shortcut to the same target will be placed in the autorun folder).

2.15 File Management

The loader's launcher provides several powerful FAT file management facilities which may be accessed either directly via shortcut keys or via the context menu. Files and folders may be renamed, deleted and copied, and many functions can work on a single item or a specified group thereof.

2.15.1 Selecting Items

In the launcher, highlighted files and folders may be selected ('marked' or 'tagged') by pressing CTRL+SPACE. A marked item is denoted by a check mark at the right-hand side of the screen.



Figure 21: Selected items

All items in a folder may be selected by pressing CTRL+A. Previously selected items can be un-selected with CTRL+SPACE (this being a ‘toggle’), and all items in a folder may be un-selected with CTRL+U.

Certain file management operations (currently ‘Copy’, ‘Cut’, ‘Delete’ and ‘Get Info’) will act upon the current highlighted item if no items are selected, or the entire selection when one or more item is selected. Note that if a folder is selected or highlighted, this implies that every folder and file in that folder will also be recursively processed by the file management operation.

2.15.2 Getting File/Folder Information

Choosing ‘Get Info’ from the context menu or pressing CTRL+I will display information about the highlighted file or folder, or the current selection of files/folders. The information includes statistics about the contents of selected folders (which are recursively scanned), and the total size of all files and folders in the selection. When only a single file is selected, the file’s attributes will be accurately displayed.



Figure 22: The ‘Get info’ function applied to a folder

2.15.3 Deleting Items

Files and folders may be deleted by choosing ‘Delete’ from the context menu or pressing CTRL+D. When no items are selected, the operation will apply to the highlighted file or folder, and when files and folders are selected, the operation will affect the selection. The contents of folders (i.e., their entire sub-tree) will be recursively erased before the top-level target folder is deleted (therefore, folders need not be empty in order to be deleted). You will be asked for confirmation before deletion is carried out.

2.15.4 Renaming an Item

A single file or folder may be renamed by choosing ‘Rename’ from the context menu or by pressing CTRL+N. The filename will be placed into edit mode and one may edit the name *in situ* in the file list. If the resulting name is legal and does not already exist in the same folder, the item will be renamed and the file list updated to reflect the change.

2.15.5 Cut, Copy and Paste

The loader has versatile cut/copy/paste facilities modelled after those in the Windows operating system (and which use the same shortcut keys). A single item or a selection may be 'cut' with CTRL+X, or copied using CTRL+C (or by choosing 'Cut' or 'Copy' respectively from the context menu). At that point, the selection is catalogued (the loader will report the number of 'discovered' items as it processes them), and once this process is complete, the selection may be pasted (with CTRL+V or by choosing 'Paste' from the context menu) anywhere else in the directory tree (pasting to the source directory is not currently permitted).

While items are being copied or moved (when 'pasted' into the target location), a progress dialogue will appear on the screen, and this reports the number of items and the amount of data processed (both as a proportion of the total size of the selection), the estimated data transfer speed, and the estimated time remaining.



Figure 23: File copying in progress

The process may be abandoned with ESC. Note that copying or moving large files may take a considerable amount of time.

Once all items have been processed, the file list will update to reflect the newly pasted items. When 'moving' files (with cut/paste), the original files and directories will be deleted once they have been written to the destination folder.

If a file of the same name already exists in the destination folder, the copier will ask the user whether they wish to delete the existing file, skip it, or skip all subsequent conflicts. If a copied or moved folder already exists in the destination folder, copied or moved items will be merged into it.

2.15.6 Creating Folders

A new folder may be created in the currently displayed folder by choosing 'New folder' from the context menu or by pressing CTRL+M. The highlight will move to the end of the file list, and the user may type the desired folder name *in situ*. If folder creation is successful, the new folder will appear in the file list prefixed by a folder icon.

2.15.7 Creating Disk Images

To create a new disk image (ATR), choose 'New image' from the context menu, or press SHIFT+CTRL+I. The loader will ask (via a dialogue) for the size/type of the new disk image, and if the user presses RETURN on the desired entry, the name of the new disk image may then be typed directly at the end of the file list. If disk image creation is successful, the new image will appear in the file list and will be ready for immediate use.



Figure 24: Creating a new disk image (ATR)

Note that new disk images are created by 'cluster pre-allocation' for the sake of speed and therefore have undefined content. This normally doesn't matter with disk images (since they will commonly need to have a file system written to them anyway), but it's worth bearing in mind that a new image might actually contain a lot of 'junk' in its sectors.

2.15.8 Changing Item Properties

To change an item's properties, select 'Set properties' from the context menu, or press SHIFT+CTRL+P. A pop-up menu shows check-marks next to currently set properties ('Read-only', 'Hidden', 'System' and 'Archive'), and allows them to be toggled with the RETURN key. Finally, selecting 'Apply' will cause the selected attributes to be applied to the item.



Figure 25: Setting item properties

2.15.9 Image and Partition Mounting

Several methods exist for mounting disk images and APT partitions. Pressing SHIFT+CTRL+SPACE assigns the next logical drive number to a partition or disk image, displays the drive number next to the filename, and moves the selection bar to the next entry. Pressing SHIFT+CTRL+SPACE on a partition or image already mounted unmounts it, removing the drive number.

To manually specify the drive number on a highlighted image file, select 'Mount' from the context menu, or press TAB or CTRL+RETURN, or use with the joystick, use BUTTON+UP. A pop-up list of drive numbers will appear, and you can select one with the cursor keys or the joystick.



Figure 26: Assigning a drive number to an image

Press ESC to cancel drive number selection. Selecting 'Off' from the list will un-mount an already mounted image.

2.15.10 Map Files and Disk Flipping

Sometimes it's necessary to mount multi-disk ATR sets and tagging each file individually can be a tedious process. 'Map' files make this easier by automating the mounting procedure. A map file is simply a plain DOS/Windows/Unix text file containing EOL terminated lines comprising a drive number, optional path and filename:

Dd:[path]filename.ext

'Dd:' represents the target drive number on which the image will be mounted, and the path/filename of the image file should immediately follow on the same line. For example, you might create a file called 'Ballyhoo.map' and add the following two lines:

```
D1:Ballyhoo (s1).atr
D2:Ballyhoo (s2).atr
```

When you press Return on 'Ballyhoo.map', the loader will mount the two ATRs on the specified drives and – if successful – reboot the Atari (if an error occurs, a message will be displayed and the computer will not be rebooted). In this case, the ATRs must reside in the

same folder as the map file. If the ATR files reside in a different folder to the map file, include the path:

D1:\ATRs\Demos\Sweet Illusions (s1).atr

D2:\ATRs\Demos\Sweet Illusions (s2).atr

Either the right wedge (>) or backslash (\) characters may be used as path delimiters, and the path will be relative from the currently logged folder unless the path begins with a path delimiter (which implies an absolute path starting from the root directory).

Once the ATRs are mounted, the Atari will boot from drive 1. In the case of 'Ballyhoo (s1).atr' and 'Ballyhoo (s2).atr', when the software requests its second disk, pressing the SIDE/ATR swap button prior to accepting acknowledging the disk swap request will move 'Ballyhoo (s2).atr' from D2: to D1:, and 'Ballyhoo (s1).atr' from D1: to D2:. The SIDE ATR swap button *rotates* the entire ATR set down by one place in the drive mappings, and wraps around the ATR on the lowest drive number to the vacated highest slot.

Take a set of three ATRs as an example. When mounted, the assignments are thus:

D1:Image1.atr

D5:Image2.atr

D6:Image3.atr

After the first disk rotation, the mount points are as follows:

D1:Image2.atr

D5:Image3.atr

D6:Image1.atr

A second disk rotation results in:

D1:Image3.atr

D5:Image1.atr

D6:Image2.atr

As can be inferred, once the number of rotations equals the number of mounted ATRs, the assignments will return to their original positions. Hard disk partitions (which may be used freely with mounted ATRs) are unaffected by the rotation process and remain fixed.

Image rotation can also be accomplished from the loader itself via the 'Swap ATRs' option on the 'Tools' menu.

Should you wish to execute a MAP file without an instant reboot, press CTRL+RETURN instead. The volumes will be mounted but no further action will be taken.

2.15.11 BASIC/SDX Enable Metatags

Executables and ATRs which require internal BASIC to be enabled on a 'one off' basis (assuming the loader is set to disable BASIC during normal use) may specify their dependency on BASIC by including the [BASIC] metadata tag anywhere in the FAT filename. For example, the following ATR will always enable internal BASIC upon booting (if launched via the Return key):

BASIC Games [BASIC].atr

Note that the BASIC metatag on an ATR file is not observed if the volume is mounted via tagging (using CTRL+SPACE) or via the TAB key.

Another supported metatag is [SDX]. If included in an image filename, this tag will cause SpartaDOS X to remain enabled when the loader automatically reboots to launch the image.

2.15.12 CF/SD Card Hot-Swapping

You can hot-swap CF/SD cards while in the loader. All disk image mounts will be removed following a card swap (although a mounted cartridge image will remain intact). When no media is inserted, the loader will prompt the user to insert an SD card. When a new card is inserted, it is immediately catalogued by the loader and the partition or launcher menus displayed (depending on the number of partitions on the card and whether the loader recognises the 'last accessed' partition on a previously inserted card and opens it automatically).

2.16 Error Messages

The following is a list of error messages which may be produced by the SIDE3 loader, along with their error codes and a brief explanation of each error condition.

128 \$80 Break abort

The BREAK key was pressed during an IO operation.

130 \$82 Bad device

A non-existent device specifier was supplied.

138 \$8A Timeout

The device failed to respond.

139 \$8B NAK

The device failed to acknowledge the command.

148 \$94 Unrecognized file system

The loader does not recognize the file system.

150 \$96 Bad path

The specified path does not exist or contained illegal characters.

151 \$97 File exists

An attempt was made to create a file or directory with a name which already exists in the same directory (this can also happen when renaming files and directories).

162 \$A2 Disk full

The FAT volume is full. If this occurs during a copy operation, the target file will be truncated in size.

163 \$A3 Bad wildcard

A wildcard character was used in a filename in a context where wildcards are not allowed (for example, when creating a new folder).

165 \$A5 Bad name

The filename contains invalid characters.

167 \$A7 Directory not empty

An attempt was made to erase a directory which is not empty (this would suggest that the directory contained files or folders which could not be deleted because their write-protect attributes were set).

170 \$AA Not found

The referenced file does not exist.

181 \$B5 File corrupt

The file system or a file's cluster chain has become corrupted.

179 \$B3 Memory full

The loader's directory buffer is full.

200 \$C8 Bad ATR

An attempt was made to mount an ATR disk image whose header is corrupt or invalid.

201 \$C9 ATR size

An attempt was made to mount an ATR disk image of an unsupported size (this error may be produced when using SIDE3's soft-OS ATR support).

202 \$CA No free slot

This error is produced when an attempt is made to mount more than four ATR disk images simultaneously when using SIDE3's soft-OS ATR support.

203 \$CB Bad density

An attempt was made to mount an ATR of an invalid density.

204 \$CC Invalid file

An attempt was made to mount media of an invalid format.

208 \$D0 Idle state

Low-level SD card error.

209 \$D1 Erase reset

Low-level SD card error.

210 \$D2 Bad command

Low-level SD card error.

211 \$D3 Command CRC error

Low-level SD card error.

212 \$D4 Erase Sequence error

Low-level SD card error.

213 \$D5 Address error

Low-level SD card error.

214 \$D6 Parameter error

Low-level SD card error.

235 \$EB CRC error

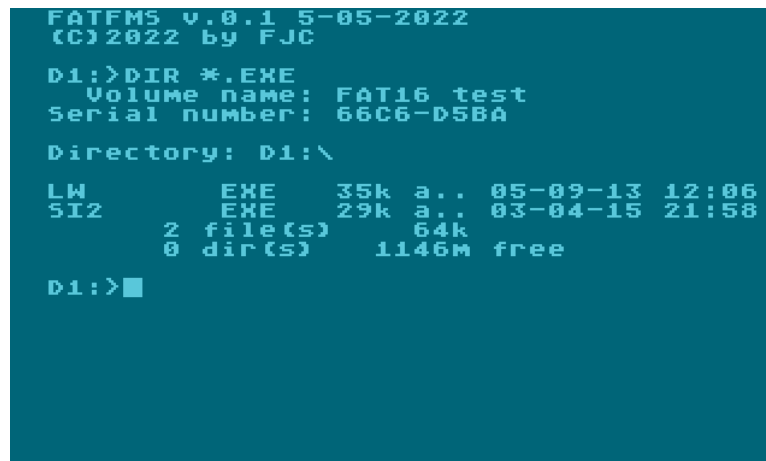
Low-level SD card error.

237 \$ED Write error

Low-level SD card error.

3 The Command Processor

FATFMS is a command-line driven DOS which requires you to type commands via a 'command-line interface' (CLI). The command processor is a separate module which is loaded into the computer's memory when you first boot the system, or when you enter DOS by leaving the running application (for example, by typing 'DOS' in BASIC). The command processor module (CMD.COM) must therefore be present on the storage volume in order for it to be accessible by DOS.



```

FATFMS V.0.1 5-05-2022
(C)2022 by FJC

D1:>DIR *.EXE
      Volume name: FAT16 test
      Serial number: 66C6-D5BA

Directory: D1:\

LW      EXE      35k  a..  05-09-13  12:06
5I2      EXE      29k  a..  03-04-15  21:58
      2 file(s)      64k
      0 dir(s)      1146M free

D1:>
  
```

Figure 27: The command processor

3.1 Command Prompt

The command processor displays the path of the current working directory (including the drive number), followed by '>' before waiting for the next command.

3.2 Drives

FATFMS uses standard 'Dd:' drive identifiers throughout (where 'd' is an optional drive number between 1 and 9). In this documentation, 'volume' and 'drive' may be considered as interchangeable terms.

3.3 Filenames

Filenames follow the usual '8.3' convention (a filename of up to eight characters, and an optional extension of up to three characters, preceded by a period). Since FATFMS uses the FAT file system, permissible characters are:

Letters 'A'-'Z' (uppercase)

Digits '0'-'9' (filenames may begin with a digit)

In addition, the following characters are allowed:

! # \$ % & ' () - ^ _ @

3.4 Long Filename (LFN) Aliases

Some filenames may appear on the Atari with the underscore ('_') character in unusual locations. This is because the underscore is used as a substitute for any character which

doesn't exist in the Atari character set. This includes the tilde ('~') character which is used in the 'short aliases' of files which also have a long filename. You can still refer to these files on the Atari by simply by using an underscore in the corresponding position of the file specification.

Care should be taken when deleting or in any way changing such 'alias' files, since they can easily become disengaged from their corresponding long filenames (which will never appear in directories limited to short filenames). For example: if you delete a file with a long filename via its short alias, the long filename data will remain in the directory, but completely 'stranded' from its short alias. As a result, the space consumed by the long filename data will be inaccessible until the volume has been processed by SCANDISK or a similar utility.

3.5 Wildcards

The '?' and '*' characters have special meaning in a file specification. '?' matches any single character, while '*' matches all characters in the rest of the filename or file extension.

3.6 File Attributes

FATFMS caters for the following file attributes:

- A – Archive
- H – Hidden
- P – Protected
- S – Subdirectory

FAT (the filesystem used by FATFMS) has two additional attributes:

- System
- Volume

In order to preserve topical compatibility with SpartaDOS 3.x and SpartaDOS X, the 'system' attribute is largely ignored, while the 'volume' attribute is only used by the volume label entry (so does not need to be alterable by the user).

3.7 Time/Date Stamps

FATFMS mandates the presence of a system clock so that newly created and updated files can have their time/date stamps properly maintained. FAT maintains two different time stamps: created and last modified. FATFMS generally reports (in directories, etc) the last modified time stamp, although both are properly maintained (and both may be manually changed). FATFMS does not support the third kind of FAT timestamp: 'last accessed'.

3.8 Directories

FATFMS supports FAT subdirectories of unlimited size, but remember that the larger a subdirectory becomes, the longer it can take to locate and open any files which reside there. It's therefore good practice to make liberal use of subdirectories as a means of arranging and categorising your files.

In directory listings in the command processor directories are suffixed with '<DIR>'. In directory listings obtained via the CIO (for example, via BASIC), directories entries are prefixed with ':'.

3.9 Pathnames

Paths describe the 'address' of a file or directory. The '\' character acts as a delimiter between the elements of a path, and on the Atari, the '>' (included for compatibility with SpartaDOS and MYDOS) does the exact same job. If '\' or '>' appears at the beginning of a path, this means begin the search from the root directory of the volume.

Examples:

```
>TBXL>COMPILER.COM  
\DOS\DELTREE.COM  
\GAMES\ATR\YOOMP.ATR
```

To refer to a directory's parent (to 'go back up the tree'), use '..\' or the '<' character. For example:

```
DIR ..\
```

The above command displays the parent directory.

```
CD <TEST>TEMP
```

The above command changes the current directory to the 'TEMP' directory which exists in the 'TEST' directory, which in turn resides in the parent of the current working directory.

3.10 Maximum Path and Command Length

The maximum supported path length is 63 characters. The command line buffer is 64 bytes in length, so any commands, switches, paths, etc, must not exceed 63 characters.

3.11 Other Device Names

In the command processor, other devices ('E:', 'S:', 'P:', etc) may be referred to by their normal names. For example, to copy a file from the screen editor, type:

```
COPY E: TEST.TXT
```

3.12 Default Drive and Directory

The default drive (commonly 'D1:') is assumed if the drive ID is omitted from any command. To change the default drive, simply type another drive's ID on its own at the command prompt.

Each drive on the system has its own default directory which may be changed with the 'CHDIR' (or 'CD') command. By default, the current working directory of each drive is the root directory.

3.13 Volume Names

All FAT formatted volumes may have a volume name of up to eleven characters. The volume name is displayed at the start of a directory listing, by the VOL command, and may be changed (or removed) with the CHVOL command.

3.14 Volume Serial Numbers

All FAT volumes have a volume serial number, set by the formatter when the volume is first created. The volume serial number is displayed by the DIR and VOL commands and may not be changed.

3.15 Media Compatibility

Because most – if not all – modern computers and mobile devices can work with the FAT file system, FATFMS allows direct media compatibility between the Atari and the PC, Mac, Linux machine, or other appliance. Files created on the Atari are 100 per cent compatible with all devices which can read and write the FAT file system. The only compatibility issue is likely to stem from the Atari's non-standard carriage return character (\$9B), which is totally incompatible with PC/Mac/Linux CR/LF sequences.

3.16 Commands

What follows is a complete reference of the ‘internal’ and ‘external’ commands available from the command processor. Internal commands are built into said command processor, while external commands are loaded from disk when needed.

3.16.1 ATR

Purpose

Change file attributes.

Syntax

ATR [+A|H|P] [-A|H|P] [d:][path]name[.ext]

Alias

ATTRIB

Type

Internal

Description

ATR is used to set and clear attributes on files matching the file specification. The specified attributes are applied to matching files. Attributes are as follows:

- A Archive
- H Hidden (does not appear in directory listings)
- P Protected (cannot be deleted or written to)

For example, to set the protected bit and clear the archived bit on all ‘.TXT’ files, use:

ATR +P -A *.TXT

Note: the subdirectory attribute (‘S’) used in attribute scan mode can NOT be changed with this command.

Hidden files – although they do not appear in directory listings unless attribute scan mode ‘+H’ is used – can still be opened, read, and written to, and hidden subdirectories can likewise be used in path specifications. Similarly, hidden files and folders can be opened via the CIO without recourse to attribute scan mode (see the ‘Programming with FATFMS’ section).

3.16.2 BASIC

Purpose

Enter the internal BASIC interpreter.

Syntax

BASIC [d:][path][fname][.ext]

Type

Internal

Description

If BASIC is typed without a filename, control will be passed to the internal BASIC interpreter. If MEM.SAV is enabled and DOS was previously called from BASIC, any BASIC program currently in RAM will be preserved. If a filename is provided with the BASIC command, that BASIC program will be executed as soon as control is passed to the interpreter (assuming the file exists, otherwise error 170 'File not found' will be encountered).

Note the MEM.SAV facility is currently (and necessarily) rather rudimentary and does not account for such things as MEMLO changing after DOS was called but before BASIC was re-entered.

3.16.3 CAR

Purpose

Start the application cartridge, if present.

Syntax

CAR

Type

Internal

Description

If CAR is typed without a filename, control will be passed to the 'external' application cartridge. If MEM.SAV is enabled and DOS was previously called from the application cartridge, any program or data currently in RAM will be preserved.

Note the MEM.SAV facility is currently (and necessarily) rather rudimentary and does not account for such things as MEMLO changing after DOS was called but before the application cartridge was re-entered.

3.16.4 CHDIR

Purpose

Change the working directory.

Syntax

CHDIR [d:][path]

Aliases

CD / CWD

Type

Internal

Description

The current working directory in a hierarchical file system like FAT is the directory referred to whenever no path is specified. If no drive is specified, the default drive (currently always 'D1:') is assumed.

The default directory of a drive is always the root directory unless the current working directory has previously been set.

3.16.5 CHTD

Purpose

Change the time and date on matching files and directories.

Syntax

CHTD [+A|H|P|S] [-A|H|P|S] [d:][path]fname[.ext]

Type

Internal

Description

This command changes the date/time stamp on matching files to the current system date and time. By default, CHTD only changes the time-stamp on unprotected and non-hidden files, but this may be overridden by specifying attributes. A file specification must be provided since '*. *' is not assumed.

3.16.6 CHVOL

Purpose

Change the volume name on the specified drive.

Syntax

CHVOL [d:][volname]

Aliases

LABEL

Type

Internal

Description

When no volume name is provided, this command will ask for a new volume name and offer to delete the current volume name if a null string is entered (this is the only way to remove the existing name without replacing it). Otherwise – if the volume name is supplied on the

command line – it will immediately be applied.

3.16.7 CLS

Purpose

Clear the screen.

Syntax

CLS

Type

Internal

Description

This command simply clears the screen before re-displaying the command prompt.

3.16.8 COLD

Purpose

Reboot the system (by jumping through \$E477).

Syntax

COLD [/N]

Type

Internal

Description

This command supports one switch:

/N Reboot the computer with any mounted 'external' application cartridge disabled

3.16.9 COPY

Purpose

Copy files and/or directories and their contents.

Syntax

COPY [switch] [+A|H|P] [d:][path][source][.ext] [d:][path][dest][.ext]

Type

Internal

Description

This command supports several option switches.

Switches:

- /B- backup mode (copy files changed since last backup)
- /C- confirmation mode (confirm every file copy)
- /D - do not preserve date and time

- /I- ask before overwriting a file
- /K- set the archive attribute on the original file after copying
- /M- delete the source file/directory (move)
- /N - skip existing destination entries
- /Q- suppress messages (except error messages)
- /R- dig recursively into subdirectories
- /V- summary (number of files and directories copied)

3.16.10 DATE

Purpose

Display and optionally set the current system date.

Syntax

DATE

Type

Internal

Description

Once the command has displayed the current date, it requests the input of a new date. To keep the current date, press RETURN at the prompt without entering a new date.

3.16.11 DELTREE

Purpose

Delete entire directory trees.

Syntax

DELTREE [d:]path

Type

External

Description

This command requests confirmation before proceeding, owing to its destructive nature. The entire sub-tree in 'path' will be deleted. If the command fails and results in a 'Directory not empty' error, check for hidden or protected files or directories.

3.16.12 DIR

Purpose

Display a directory.

Syntax

DIR [+A|H|P|S] [-A|H|P|S] [d:][path][fname][.ext] [/S]

Type

Internal

Description

'DIR' invoked without any switches displays the names of all files and directories in the current working directory. At the top of the directory, the volume name and serial number are displayed, followed by the directory's path. Below that, the output shows – left to right – the filename, filename extension, file size (or '<DIR>' if the entry is a subdirectory), attributes (archived, hidden and protected), and the last modified date and time. Finally, at the end of the directory listing, the number of displayed files, their cumulative size, the number of displayed directories and the free space on the disk are shown.

```

D1:\TBXL>DIR
Volume name: FAT16 test
Serial number: 66C6-D5BA

Directory: D1:\TBXL

.                <DIR>    ...  26-12-21  17:34
..               <DIR>    ...  26-12-21  17:34
TBXL             COM      18k  a..  25-02-22  19:43
COMPILER        COM 9933b  a..  18-10-21  12:47
RUNTIME         COM      11k  a..  26-02-22  11:56
  3 file(s)      39k
  2 dir(s)       1152M free

D1:\TBXL>

```

Figure 28: Output of the DIR command

File sizes, the cumulative size of displayed entries, and the free space on the volume are displayed with a maximum of four digits. Thus, 9999k is the highest value in kilobyte units ('k') which can be displayed before the unit changes to megabytes ('m'), and so on.

Only three attributes are displayed in the directory (archive, hidden and protected) even though FAT also supports the 'system' attribute. To display a full list of file attributes, see the 'INFO' command.

The modification time stamp is displayed in 24-hour format without seconds. To see the full time stamp (and the creation date/time), see the 'INFO' command.

To narrow the results to match a particular file mask or to show a directory in a different location on the disk, simply provide the device/path/filespec after the DIR command.

One may further narrow the directory listing to include only those entries with or without a certain file attribute. The attributes are:

- A Archive
- H Hidden (does not appear in directory listings)
- P Protected (cannot be deleted or written to)
- S Subdirectory (the only attribute which may not be changed)

To specify that files should have a particular attribute or set of attributes set, preceded with a '+' symbol. To specify that files should *not* have one or more attributes set, prefix with '-'. For example, to list all directory entries which are not protected and have the archive bit cleared:

DIR +S -PA

To display all files with the 'TXT' extender and ignore any matching directories:

DIR -S *.TXT

Note that '-H' is implied if no attribute mask is provided.

If no filespec is provided, '*.*' is assumed. Paths (if lacking a file specification) should be terminated with '>' or '\'. For example, to list all files in the 'BASIC' directory:

DIR BASIC>

Currently the only switch supported by DIR is '/S', and this will cause the directory to be recursive (listing the contents of every folder at or below the level of the current working directory). To list all files and directories, type:

DIR /S

Note that directory entries with the 'volume' attribute set are never displayed, and entries with the system attribute set are currently always displayed.

Note also that a short delay may be experienced with FAT16 volumes before the free space information at the end of the directory is displayed (if the system had not previously established the free cluster count on the drive before the DIR command was issued).

3.16.13 ERASE

Purpose

Delete the specified file(s).

Syntax

ERASE [d:][path]filename[.ext]

Alias

DEL, DELETE

Type

Internal

Description

Wildcards may be used to delete multiple files, but if '*.*' is used, the command will prompt the user for confirmation that they want to delete every file in the directory.

Hidden files and files with the protected bit set cannot be erased until those attributes are first cleared (with the ATR command).

3.16.14 EXIT

Purpose

Leave the command processor and return to the SIDE3 Loader.

Syntax

EXIT

Type

Internal

Description

This command hands control to the SIDE3 Loader.

3.16.15 HELP

Purpose

Display a command reference.

Syntax

HELP

Type

Internal

Description

This command displays brief usage information for all internal commands.

3.16.16 INFO

Purpose

Obtain detailed information about a specified file.

Syntax

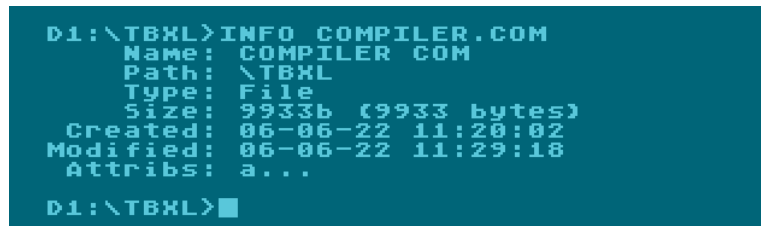
INFO [d:][path]filename[.ext]

Type

Internal

Description

This command displays the name, path, type, size, creation date/time, modification date/time, and attributes of the specified file. Unlike DIR, this command displays the 'seconds' field of the time-stamps, and the 'system' attribute.



```
D1:\TBXL>INFO COMPILER.COM
      Name:  COMPILER.COM
      Path:  \TBXL
      Type:  File
      Size:  9933b (9933 bytes)
Created:  06-06-22  11:20:02
Modified:  06-06-22  11:29:18
Attribs:  a...
D1:\TBXL>
```

Figure 29: Output of the INFO command

If wildcards are used, the command will simply display information about the first matching file or directory.

3.16.17 LOCK

Purpose

Set the protect attribute on the specified files or directories, making them read-only.

Syntax

LOCK [d:][path]filename[.ext]

Type

Internal

Description

This command is provided as a convenience and does the exact same job as the ATR command when the '+P' attribute is specified. Once locked, a file or directory cannot be deleted, renamed, or opened for write.

3.16.18 MEM

Purpose

Display memory usage information.

Syntax

MEM

Type

Internal

Description

This command displays the current system MEMLO and MEMHI values. MEMLO will commonly point to the first free location after the DOS sector buffer(s) (\$2000), while MEMHI points to the screen memory address (\$BC1F when no external cartridge present, \$9C1F with 8K external cartridge present).

3.16.19 MEMSAV

Purpose

Enable or disable the 'MEM.SAV' feature.

Syntax

MEMSAV ON|OFF

Type

Internal

Description

The 'MEM.SAV' facility caches application memory (including the upper half of page zero) in a file on disk, such that DOS can be called from BASIC, the application cartridge, or even a disk-loaded application without losing the contents of user RAM. When MEM.SAV is active,

for example, typing 'DOS' in BASIC, copying files in the command processor, and then typing 'BASIC' to re-enter internal BASIC will result in any BASIC program in memory remaining intact. To re-enter a disk-loaded application in this manner, the 'RUN' command should be used (with the run address of the program, which must be known). For example, when Turbo BASIC XL is loaded, typing 'RUN \$2080' will re-enter TBXL with the user program intact when MEM.SAV is active.

3.16.20 MKDIR

Purpose

Create a subdirectory.

Syntax

MKDIR [d:][path]

Alias

MD

Type

Internal

Description

The command creates a new, empty directory at the specified path, on the specified drive (the default drive and path are assumed if only the new directory name is provided). For example, to create a new directory on the current drive in the current folder, type:

MD NEWDIR

To create a folder called 'PROGRAM' in the 'TBXL' folder which is in the root directory of the current drive, use:

MKDIR >TBXL>PROGRAM

Note that files and folders may not share the same name, so if a file 'TEST' exists, 'MKDIR TEST' will fail with a 'File exists' error.

3.16.21 RENAME

Purpose

Change the name of one or more files.

Syntax

RENAME [d:][path]name[.ext] newname[.ext]

Alias

REN

Type

Internal

Description

Wildcards may be used in the original and new filenames, but a device/path may only be specified for the original name. Blank filenames are not allowed. To rename all 'TXT' files to 'BAK' files, for example:

```
RENAME *.TXT *.BAK
```

To give all files beginning 'A' a 'FOO' extender:

```
RENAME A*.* *.FOO
```

Note that if the operation would result in two files with the same name (or a file with the same name as a directory), a 'File exists' error will be generated and the file's name will not be changed.

3.16.22 RENDIR

Purpose

Change the name of a directory.

Syntax

```
RENDIR [d:][path]oldname[.ext] newname[.ext]
```

Type

Internal

Description

This command works exactly like the rename command, but applies to directories instead of files, and will not rename multiple directories when wildcards are used (only the first matching directory will be renamed).

3.16.23 RMDIR

Purpose

Delete and empty subdirectory.

Syntax

```
RMDIR [d:]path
```

Alias

RD, DELDIR

Type

Internal

Description

The last element in the path will be deleted if it exists and is empty. An empty directory contains only the current directory ('.') and parent directory ('..') entries. For example:

```
RMDIR D1:>TBXL>PROGRAM
```

The directory 'PROGRAM' will be removed, providing it contains no files or other directories.

If a directory appears empty but still cannot be deleted, check for hidden files. If no hidden files exist, the directory may contain entries for files which were opened but never closed. A disk-checking utility such as 'SCANDISK' should be used to remove such broken directory entries.

Note if an attempt is made to delete the current working directory, a 'File exists' error will be generated.

3.16.24 RUN

Purpose

Transfer control to code at the specified memory address or the system RUN address.

Syntax

RUN [\$][address]

Type

Internal

Description

Typed without any parameters, this command transfers control to the address at RUNAD (\$2E0). Alternatively, an address can be specified in decimal or hexadecimal.

When MEM.SAV is active, this command can be used to re-enter Turbo BASIC XL by typing:

RUN \$2080

User memory will be re-instated (from the cache in the 'MEM.SAV' file) and the program the user was working on before DOS was called will reappear in memory.

3.16.25 SPARTA

Purpose

Enable or disable SpartaDOS impersonation.

Syntax

SPARTA [ON|OFF]

Type

Internal

Description

By default, FATFMS presents as a unique file system driver, and therefore cannot be identified by applications as a known 'command line' DOS. Thus applications written to exploit the command line (or while exit directly to DOS without waiting for a keystroke, in the knowledge that the command line DOS will not clear the screen and present a menu after the execution of every program) in the presence of SpartaDOS, BW-DOS, OS/A+, etc, if they run at all, will believe themselves to be running under a menu-based DOS with no command line buffer.

The SPARTA command – with the 'ON' parameter – causes FATFMS to present itself as SpartaDOS 3.x (via the SpartaDOS signature and version bytes at \$7xx). While this enables many programs which support SpartaDOS to function properly and even avail themselves of command line arguments, the feature will cause problems with other programs which – believing they are running under SpartaDOS – attempt to call kernel or OSRAM vectors which do not exist. 'Your mileage may vary' as a result.

For example, when SpartaDOS mode is enabled, KMK's 'RWTEST' program does not wait for a keystroke after completion, and works perfectly. RWCRC, on the other hand, will crash, since it presumably attempts to call a page seven or 'COMTAB' function which is not supported by FATFMS.

SPARTA typed without any parameters simply reports the current state of SpartaDOS mode.

3.16.26 UNLOCK

Purpose

Clear the protect attribute on the specified files or directories, making them read/write.

Syntax

UNLOCK [d:][path]filename[.ext]

Type

Internal

Description

This command is provided as a convenience and does the exact same job as the ATR command when the '-P' attribute is specified. Once unlocked, a file may be deleted, renamed or overwritten. An unlocked directory may be deleted (when empty) or renamed.

3.16.27 VER

Purpose

Display DOS and (optionally) command processor version information.

Syntax

VER [/X]

Type

Internal

Description

Without the 'X' switch, this command displays the DOS version and copyright information. With the 'X' switch, the version and revision date of the command processor is also displayed.

3.16.28 VOL

Purpose

Display information about the specified volume (drive).

Syntax

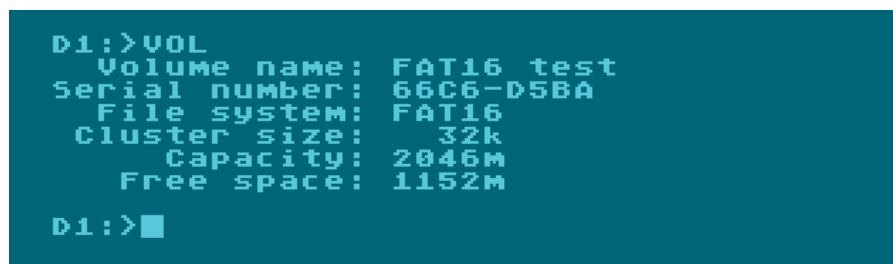
VOL [d:]

Type

Internal

Description

This command displays the volume name, serial number, file system (FAT12/16/32), cluster size, capacity and free space on the specified volume.



```
D1:>VOL
Volume name:  FAT16 test
Serial number: 66C6-D5BA
File system:   FAT16
Cluster size:  32k
Capacity:      2046M
Free space:    1152M

D1:>■
```

Figure 30: Output of the VOL command

Note that a short delay may be experienced with FAT16 volumes before the information is displayed (if the system had not previously established the free cluster count on the drive).

4 Programming with FATFMS

4.1 Accessing DOS facilities from BASIC

Many FATFMS functions may be accessed from BASIC, machine code, or other programming languages like Action!, Turbo BASIC XL, Altirra BASIC, etc. What follows is a complete list of CIO functions (including XIO commands) in a format accessible by Atari BASIC. Assembly language programmers or those using another interpreter or compiled language should not find it difficult to convert the BASIC calling methods to something usable in their chosen dialect.

In the following list, 'IOCB' refers to an 'Input/Output Control Block' numbered from 0 to 7. Since IOCB #0 is usually used for the screen editor, file operations commonly employ IOCBs 1 onwards.

'd:' refers to a drive's (optional) unit number, 'path' refers to a file's pathname, and 'filename.ext' refers to a file's name, as per the naming conventions described in section 2.

4.1.1 Open File

Purpose

Open a file for read/write.

SpartaDOS Compatibility

Partial

Syntax

OPEN #IOCB,aux1,aux2,"Dd:[path]filename.ext"

Description

This command opens the specified file on disk. Aux1 describes the access mode. Below is a list of value values for aux1. Unless otherwise stated, aux2 should be 0.

- 4 Open the file in read-only mode
- 6 Open a formatted directory for read. Only the 'short' (Atari DOS 2.x) style formatted directory is currently available owing to space constraints.
- 8 Open the file in write-only mode.
- 9 Open the file in append mode. Data will be written to the end of the file if it already exists, and if the file doesn't exist, it will be created.
- 12 Open the file in update mode. Reading and writing to the file are permitted in this mode (although the file size may not be extended).

Example

This short BASIC program will read a formatted directory and display it on the screen:

```
10 DIM A$(64)
20 OPEN #1,6,0,"D1:*.*)"
30 TRAP 60
40 INPUT #1;A$:PRINT A$
```

```
50 GOTO 40
60 CLOSE #1
```

If the file is opened for write and does not exist, it will be created. If a file opened for write already existed, it will be replaced. If a file is opened for read and it doesn't exist, a 'File Not Found' error will be generated.

4.1.1.1 Accessing the Raw Directory

In a similar manner to SpartaDOS, setting bit 4 of aux1 (i.e. adding 16 to the base value) allows access to the 'raw directory', which is the unformatted data straight from the disk. Unfortunately, the 'raw' FAT directory structure is not directly compatible with the raw directory format of SDFS, but the method of accessing the raw directory is at least more or less the same.

Needless to say, extreme care should be exercised when accessing the raw directory in write mode, since a programming error could result in total obliteration of the entire volume (especially when working with the root directory).

The command processor happens to use raw directory access when producing the 'long' formatted directory output, as well as when changing the volume label. In raw directory mode, directories act exactly like files and may be accessed using the same modes. The only special case is the root directory of a FAT16 volume, which has a fixed size and may not be extended.

Raw FAT directory entries are 32 bytes in size. When reading raw directories, it can be expedient for reasons of performance to read the directory in 512 byte chunks. This will enable DOS to 'burst' every sector of the directory into memory, rather than transferring the data one byte at a time via a buffer.

4.1.1.2 Attribute Scan Mode

The open operation may be placed in 'attribute scan mode' by adding 64 to aux1 (i.e. setting bit 6). In scan mode, aux2 is then used to specify the desired attribute mask. The desired values from the following table should be added together and the result placed in aux2:

Protected	1	Unprotected	16
Hidden	2	Not hidden	32
Archived	4	Not archived	64
Subdirectory	8	Not a subdirectory	128

Only files matching both conditions (i.e. with the desired attributes and without the excluded attributes) will match.

Note that – unlike SpartaDOS – under FATFMS, files with the 'hidden' attribute set may be opened without recourse to attribute scan mode (all the 'hidden' attribute does is prevent the filename appearing in directory listings).

Note: SpartaDOS formatted directory options and modifiers are not currently supported.

4.1.2 Rename File(s)

Purpose

Change the name of one or more files.

SpartaDOS Compatibility

Full

Syntax

```
XIO 32,#IOCB,0,aux2,"Dd:[path]oldname[.ext] newname[.ext]"
```

Description

The file or files matching 'oldname.ext' will have their name(s) changed to 'newname.ext'. The IOCB must be closed before calling this function. Wildcards are supported in both the original and new filenames.

If aux2 = 0, this function renames files. If aux2 = 128, the function acts like the RENDIR command and renames directories.

Note that's impossible to perform a rename operation which would result in one or more files or directories having the same name (a 'File exists' error will be generated and the rename operation will be aborted).

4.1.3 Erase File(s)

Purpose

Delete one or more files.

SpartaDOS Compatibility

Full

Syntax

```
XIO 33,#IOCB,0,0,"Dd:[path]filename[.ext]"
```

Description

The specified file or files (unless write-protected) will be erased from the volume. The IOCB should be closed prior to this operation. Wildcards are supported (care should be exercised here, lest every file in a directory is accidentally deleted).

4.1.4 Protect File(s)

Purpose

Set the write-protect flag on one or more files.

SpartaDOS Compatibility

Full

Syntax

```
XIO 35,#IOCB,0,0,"Dd:[path]filename[.ext]"
```

Description

Files affected by this function will be rendered read-only until the write-protect bit is again cleared (either by the 'Set Attributes' function, or the 'Unprotect' function). Protected files may not be erased, changed, overwritten, or renamed. The IOCB should be closed before this function is called.

4.1.5 Unprotect File(s)

Purpose

Clear the write-protect flag on one or more files.

SpartaDOS Compatibility

Full

Syntax

```
XIO 36, #IOCB, 0, 0, "Dd:[path]filename[.ext]"
```

Description

This function clears the write-protect flag on previously protected files, allowing them to be renamed, overwritten, or deleted. The IOCB should be closed before this function is called.

4.1.6 Set File Position (Point)

Purpose

Set the position of the file pointer.

SpartaDOS Compatibility

Full

Syntax

```
X=POS
Y=0
POINT #IOCB,X,Y

or

A=INT(POS/65536)
B=INT((POS-A*65536)/256)
C=POS-A*65536-B*256
POKE 844+IOCB*16,C
POKE 845+IOCB*16,B
POKE 846+IOCB*16,A
XIO 37, #IOCB, aux1, aux2, "Dd:"
```

Description

Like SpartaDOS and derivatives (but unlike Atari DOS 2.x, MYDOS, etc), FATFMS allows direct random access to file contents via a relative file pointer. When the file pointer is zero, it is at the start of the file, ready to read or write the first byte in the file. When the file pointer's value equals the size of the file, it is at the end of the file, and the next read or write will

result in enlargement of the file or an 'End of File' error (depending on the access mode used).

A limitation with Atari BASIC and several other interpreters means that the first method shown above will only work with file pointer positions between 0 and 32767. To set the file pointer to a larger value, the second method is required (aux1 and aux2 must hold the same values they had when the file was opened).

Turbo BASIC XL, Action!, and many other languages do not have this limitation and the built-in POINT command may therefore be used freely with values greater than 32767. For example:

```
Y=INT(POS/65536)
X=POS-Y*65536
POINT #IOCB,X,Y
```

Unlike SpartaDOS X, FATFMS does not support 'sparse files', and for this reason, it is not possible to position the file pointer beyond the end of the open file. The FAT file system indeed has no provision whatsoever for sparse files.

Note that although the FAT file system can support file sizes up to 4GB, the current implementation of NOTE/POINT in FATFMS only supports file pointer manipulation up to a 16MB offset.

4.1.7 Get File Position (Note)

Purpose

Obtain the current position of the file pointer.

SpartaDOS Compatibility

Full

Syntax

```
NOTE #IOCB,X,Y
POS=X+65536*Y
```

Description

Since NOTE in Atari BASIC does not suffer from the limitations of the POINT function described in the previous section, the XIO method is not required and one may use the method shown above to obtain the file pointer position regardless of the offset.

4.1.8 Get File Length

Purpose

Obtain the length of an open file.

SpartaDOS Compatibility

Full

Syntax

```
XIO 39,#IOCB,aux1,aux2,"Dd:"
A=PEEK(844+IOCB*16)
B=PEEK(845+IOCB*16)
C=PEEK(846+IOCB*16)
LENGTH=A+B*256+C*65536
```

Description

The function will return the size of the open file (up to 16MB). Aux1 and aux2 should have the same values used when the file was initially opened.

4.1.9 Load Binary File

Purpose

Load and execute a binary file.

SpartaDOS Compatibility

Partial

Syntax

```
XIO 40,#IOCB,4,0,"Dd:[path]fname.ext"
```

Description

Loads and runs the specified binary file. If no run address is provided, control is transferred to the address of the first loaded segment. The IOCB should be closed before this function is called.

Note: SpartaDOS aux2 modifiers for this command (do not run, etc) are not currently supported.

4.1.10 Change Default Drive and Directory

Purpose

Change the default drive and directory in a single operation.

SpartaDOS Compatibility

Full

Syntax

```
XIO 41,#IOCB,0,0,"Dd:[path]"
```

Description

This function will set the working directory on the specified drive, *and* make the specified drive the default drive (i.e. the drive number assumed when no unit number is supplied).

4.1.11 Create a Directory (MKDIR)

Purpose

Create a new, empty subdirectory.

SpartaDOS Compatibility

Full

Syntax

```
XIO 42,#IOCB,0,0,"Dd:[path]newdir"
```

Description

This function creates a new, empty directory at the specified path (which must be valid, 'newdir' being the name of the new directory to be created). In fact the new directory is not really 'empty', since it will contain two entries which cannot be deleted or changed: the 'current' ('.') and 'parent' ('..') directory shortcuts. A directory containing only those two entries is deemed 'empty' (and of course the root directory contains only the 'current' directory shortcut).

The IOCB should be closed before this function is called.

4.1.12 Delete a Directory (RMDIR)

Purpose

Delete an empty directory.

SpartaDOS Compatibility

Full

Syntax

```
XIO 43,#IOCB,0,0,"Dd:[path]dirname"
```

Description

This function deletes the specified directory, providing it contains none but the '.' and '..' entries. The IOCB should be closed beforehand.

4.1.13 Change Current Directory (CHDIR)

Purpose

Change the current working directory.

SpartaDOS Compatibility

Full

Syntax

```
XIO 44,#IOCB,0,0,"Dd:path"
```

Description

This function sets the directory which will be used when no path is specified. The IOCB should be closed beforehand.

4.1.14 Set Attributes (ATR)

Purpose

Set or clear the protected, hidden, and archived status of files or directories.

SpartaDOS Compatibility

Full

Syntax

```
XIO 49,#IOCB,aux1,aux2,"Dd:filename[.ext]"
```

Description

This function modifies file/directory attributes. Wildcards are permitted. Aux1 specifies the attributes to be set/cleared in the target, and aux2 is used to specify the files affected (similar to 'attribute scan' mode with regard to the 'open' command). Add the values of the desired attributes to be set/cleared, and place the result in aux1:

Protect	1	Unprotect	16
Hide	2	Unhide	32
Set archive	4	Clear archive	64

Aux2 should be set as per the 'attribute scan mode' of the 'open' command. Add the relevant values together and place the result in aux2 to specify the sub-set of files which will be affected (files will have to satisfy the filename mask and the specified attribute criteria).

Protected	1	Unprotected	16
Hidden	2	Not hidden	32
Archived	4	Not archived	64
Subdirectory	8	Not a subdirectory	128

As an example, to set the archive bit on all protected files (excluding subdirectories) on drive 1 with a 'TXT' extender, use:

```
XIO 49,#1,4,129,"D1:*.TXT"
```

The IOCB should be closed beforehand.

4.1.15 Set Time Stamp to Current Date and Time (CHTD)

Purpose

Set a file or directory's time/date stamp to reflect the current system time/date.

SpartaDOS Compatibility

None (not supported by SpartaDOS)

Syntax

```
XIO 51,#IOCB,aux1,0,"Dd:[path]filename[.ext]"
```

Description

This function changes the time/date of a file or directory to the current system time and

date. Since FAT supports 'created' and 'modified' time stamps, aux1 may be used to specify which time stamp should be modified. By default, (aux1 = 0), both the 'created' and 'modified' time stamps are changed, but either can be left unchanged by using the following values in aux1:

- 0 Change both 'created' and 'modified' time stamps
- 64 Do NOT change the 'modified' time stamp
- 128 Do NOT change the 'created' time stamp

The IOCB should be closed beforehand.

Note: No equivalent CIO function exists in SpartaDOS.

Note: The next two functions are not available through XIO calls and therefore must be accessed directly via the CIO. BASIC and assembly language examples are provided (these examples are based on examples provided in the SpartaDOS X manual by DLT).

4.1.16 Get Disk Information (VOL)

Purpose

Obtain information about a disk volume.

SpartaDOS Compatibility

None

IOCB Data

iccom = 50
 icbal = low byte of 'Dd:' address
 icbah = high byte of 'Dd:' address
 icbll = low byte of output buffer address
 icblh = high byte of output buffer address

Description

Although this function is topically similar to command 47 (CHKDSK) under SpartaDOS, compatibility with that command is impossible for a number of reasons, not least the fact that FAT geometry data, the volume name, etc, cannot be squeezed into a compatible data structure. Therefore, a different (proprietary) command number has been used. No equivalent to command 47 is provided, and that command will generate a 'Bad command' error.

CIO Output

The output is returned in a 32-byte buffer whose format is as follows:

Offset	Description	Size (bytes)	Values
0	File system ID	1	0-2
1	Number of 512-byte sectors per cluster	1	1-128
2	Total number of clusters on disk	4	-

Offset	Description	Size (bytes)	Values
6	Number of free clusters on disk	4	-
10	Volume name	11	-
21	Volume serial number	4	-
25	Reserved	7	-

Note: at the time of writing (owing to space constraints), only *FAT Width*, *Sectors Per Cluster*, *Clusters on Disk* and *Free Clusters on Disk* are actually implemented.

4.1.17 Get Current Directory Path

Purpose

Get the path from the root of a drive to the current directory or specified path.

SpartaDOS Compatibility

Full

IOCB Data

iccom = 48
icbal = low byte of 'Dd:[path]' address
icbah = high byte of 'Dd:[path]' address
icbll = low byte of output buffer address
icblh = high byte of output buffer address

Example

There now follows a BASIC program which provides an example of the previous two CIO functions, along with an assembly language listing of the machine code routine used to access these functions via the CIO.

```

10 DIM CIO$(32),BUFFER$(64),DRIVE$(4),VOL(17)
20 DRIVE$="D1: ":DRIVE$(4)=CHR$(155)
30 RESTORE 50
40 FOR X=1 TO 32:READ Y:CIO$(X)=CHR$(Y
):NEXT X
50 DATA 104,104,104,10,10,10,10,170,104,104,157,66,3
60 DATA 104,157,69,3,104,157,68,3,104,157,73,3,104,157
70 DATA 72,3,76,86,228
80 REM MAIN LOOP
90 BUFFER$(1)=CHR$(0):BUFFER$(64)=CHR$(0)
100 BUFFER$(2)=BUFFER$
110 ? :? "CIO Call Demonstration"
120 ? :? "1 -> Get disk information"
130 ? "2 -> Path to current directory"
140 INPUT CHOICE
150 IF CHOICE<>1 AND CHOICE<>2 THEN GOTO 120
160 IF CHOICE=1 THEN ICCOM=50

```

```

165 IF CHOICE=2 THEN ICCOM=48
170 ? :? "Which drive";:INPUT D
175 ?
180 D=INT(D):IF D<1 OR D>9 THEN GOTO 170
190 DRIVE$(2,2)=STR$(D):IOCB=1
200 X=USR(ADR(CIO$),IOCB,ICCOM,ADR(DRIVE$),ADR(BUFFER$))
210 IF CHOICE=1 THEN GOTO 270
220 IF BUFFER$(1,1)=CHR$(0) THEN ? "Root directory":GOTO 80
230 FOR X=1 TO LEN(BUFFER$)
240 IF BUFFER$(X,X)=CHR$(0) THEN
BUFFER$(X)=">":BUFFER$=BUFFER$(1,X):POP :GOTO 260
250 NEXT X
260 ? BUFFER$:GOTO 80
270 ? "Cluster size: ";
280 SECTORS PER CLUSTER=ASC(BUFFER$(2,2))
290 ? SECTORS PER CLUSTER/2;"k"
300 TOTALCLUSTERS=ASC(BUFFER$(3,3))+256*ASC(BUFFER$(4,4))+65536*ASC(
BUFFER$(5,5))+16777216*ASC(BUFFER$(6,6))
310 TOTALBYTES=TOTALCLUSTERS*(512*SECTORS PER CLUSTER):TOTALKB=INT(TOT
ALBYTES/1024)
310 TOTALKB=INT(TOTALBYTES/1024)
320 ? " Total space: ";
330 ? TOTALKB;"k"
340 FREECLUSTERS=ASC(BUFFER$(7,7))+256*ASC(BUFFER$(8,8))+65536*ASC(B
UFFER$(9,9))+16777216*ASC(BUFFER$(10,10))
350 FREEBYTES=FREECLUSTERS*(512*SECTORS PER CLUSTER):FREEKB=INT(FREEBY
TES/1024)
360 ? " Free space: ";
370 ? FREEKB;"k"
380 GOTO 80

```

```

;origin is arbitrary since it will be in a string
ciov = $E456
iccom = $0342
icbal = $0344
icbah = $0345
icbll = $0348
icblh = $0349
*=$5000 ;or whatever
pla ;number of arguments.
Pla ;should be 0
pla ;iocb channel number
asl a ;multiply by 16 for
asl a ;proper IOCB form
asl a
asl a
tax ;in x where it belongs

```

```
pla ;0 again
pla ;command number
sta iccom,x
pla ;address of "Dx:" string
sta icbah,x
pla ;buffer address
sta icbal,x
pla
sta icblh,x
pla
sta icbll, x
jmp ciov ;all done. Jump CIO
```


4.2 FATFMS User-Accessible Data Table

In an attempt to maintain some compatibility with software written for OS/A+, DOS XL, SpartaDOS, BeweDOS, and SpartaDOS X, FATFMS provides a user-accessible data table pointed to by the OS variable DOSVEC at address \$0A. An assembly language example is included following the table.

Locations COMTAB, ZCRNAME, BUFOFF, COMFNAM and LBUF are supported by OS/A+ and DOS XL. Many 'COMTAB' variables supported by BeweDOS, SpartaDOS, and SDX lack support from FATFMS. The intention is to provide topical compatibility with basic tasks such as parsing the command line.

LSIO COMTAB-10

Pointer to the internal SIO routine (necessary for block hard disk transfers when no OS/PBI support is present).

COMTAB COMTAB+0

A 6502 JMP instruction followed by the address of the DOS entry code. JMP (\$0A) enters the command processor.

ZCRNAME COMTAB+3

A 6502 JMP instruction followed by the address of the filename 'crunch' routine. This function is used to pull the next argument from the command line, add a device identifier ("D:") if none is present, and place the result in COMFNAM, and return with the 6502 carry flag clear. If no further arguments are found when this function is called, the 6502 carry flag will be set on return. Since the 6502 has no indirect JSR instruction, the method presented in the example at the end of this section may be employed.

BUFOFF COMTAB+10

This location holds the offset into LBUF when the search for the next command-line argument will commence. Writing zero to this location will cause ZCRNAME to return the first argument on the command line the first time it is called (by default, ZCRNAME begin by returning the second argument, since the first is usually the name of the executed file).

DATER COMTAB+13

The date in DD/MM/YY format (3 bytes). Currently only updated by OPEN and CLOSE operations.

TIMER COMTAB+16

The time in HH:MM:SS format (3 bytes). Currently only updated by OPEN and CLOSE operations.

ODATER COMTAB+19

The creation date of a newly opened file or the modified date of a file when it is closed will reflect the date stored in these three bytes when TDOVER is set to \$FF before calling the CIO.

OTIMER COMTAB+22

The creation time of a newly opened file or the modified time of a file when it is closed will reflect the date stored in these three bytes when TDOVER is set to \$FF before calling the CIO.

TDOVER COMTAB+25

When this location is non-zero, the next file OPEN operation will cause the file's creation time-stamp to be populated with the contents of ODATER. If this location is non-zero prior to a CLOSE operation, the file's modified time-stamp will reflect what is stored in ODATER. The location is self-clearing after use.

COMFNAM COMTAB+33

Destination buffer for the ZCRNAME routine. This buffer will always begin with "Dd:" since the default drive is added when none is provided. Switches, etc, will always begin at COMFNAM+3. The buffer is 28 bytes long.

LBUF COMTAB+63

Input buffer for the command processor. The entire command line is stored here, and the buffer is 64 bytes long.

4.3 Error Codes

The following is a list of error codes which may be produced by the FATFMS file system driver. The error code will be returned in the 6502 Y register when the OS returns from the CIO.

3 \$03 Last byte of file reached

This is really a status code rather than an error (since it does not set the N flag when returning from the CIO), and is returned when the last byte of a file has just been read (at all other times, unless an error has occurred, \$01 will be returned in Y).

128 \$80 Break abort

The BREAK key was pressed during an IO operation.

129 \$80 Alreay open

An attempt was made to open a file which is already open.

130 \$82 Nonexistent device

A nonexistent device specifier was supplied.

131 \$83 File is write-only

An attempt was made to read from a file which is open in write-only mode.

132 \$84 Bad CIO command

The CIO was called with an invalid command code. All function codes above 13 are considered XIO commands, so will return 'No function in device handler' instead of this error.

133 \$85 Not open

IO was attempted on a file which has not yet been opened.

134 \$86 Bad file handle

The CIO was called with an invalid channel number in the X register.

135 \$87 File is read-only

A write operation was attempted on a file which is open in read-only mode.

136 \$88 End Of File

Reading was attempted beyond the end of file (EOF).

137 \$89 Truncated record

The record read overflowed the designated buffer size.

138 \$8A Device timeout

The device failed to respond.

139 \$8B Device NAK

The device failed to acknowledge the command.

140 \$8C SIO framing error

The device failed to acknowledge the command.

142 \$8E SIO overrun

The device failed to acknowledge the command.

143 \$8F SIO checksum error

The device failed to acknowledge the command.

144 \$90 Write protected or bad sector

The device failed to acknowledge the command.

146 \$92 No function in device handler

The device failed to acknowledge the command.

148 \$94 Unknown filesystem

The device failed to acknowledge the command.

150 \$96 Path not found

The specified path does not exist or contained illegal characters.

151 \$97 File exists

An attempt was made to create a file or directory with a name which already exists in the same directory (this can also happen when renaming files and directories).

152 \$98 Not binary file

The device failed to acknowledge the command.

161 \$A1 Too many open channels

The device failed to acknowledge the command.

162 \$A2 Disk full

The FAT volume is full. If this occurs during a copy operation, the target file will be truncated in size.

163 \$A3 Illegal wildcard in name

A wildcard character was used in a filename in a context where wildcards are not allowed (for example, when creating a new folder).

165 \$A5 Bad filename

The filename contains invalid characters.

166 \$A6 Range error

The filename contains invalid characters.

167 \$A7 Directory not empty

An attempt was made to erase a directory which is not empty.

170 \$AA File not found

The referenced file does not exist.

181 \$B5 File system corrupt

The file system has become corrupted.

182 \$B7 Path too long

The supplied path name is longer than 64 characters.

5 Technical Information

5.1 File Systems

The SIDE3 Loader works with FAT file systems; specifically, FAT16 and FAT32. FAT12 support was omitted for reasons of conciseness in FATFMS (which is severely restricted in terms of code space), and owing to the limitation on volume size imposed by that file system. Neither the loader nor FATFMS support the more recent exFAT file system.

5.2 DOS

FATFMS is installed from ROM by the loader whenever an executable is loaded (providing the FMS is enabled in the loader settings) or when a BASIC program is started. In these circumstances, no command processor module is installed, so calling DOS (via a JMP through \$0A) will return to the loader itself. The CLI module (which is itself a standard executable from which one can launch other programs) must be launched first in order for launched applications (then run from the command line) to return to 'DOS' in the usual manner when a JMP(\$0A) is performed.

5.3 MEM.SAV

The MEM.SAV file used by FATFMS is a crude 'dump' of the upper half of zero-page memory, and main memory from \$2000 to MEMTOP. For reasons of brevity, no checks are made for MEMTOP having changed when the cached memory is re-loaded (when the user re-enters an application cartridge or performs a run at address).

5.4 Implementation Notes

The FATFMS file system driver is implemented in just under 5KB of machine code residing at (\$0Bxx to \$1DFF). The memory map is as follows:

Memory region	Contents
\$0700-\$0Bxx	User-accessible data table, buffers, low-level IO code
\$0Bxx-\$1DFF	File system driver
\$1E00-\$1FFF	DOS sector buffer
\$2000-MEMTOP	Application space

From the memory map, it should be clear that there is currently no room for the installation of additional drivers in low memory (since applications commonly load no higher than \$2000). Applications which load lower than \$2000 will not work.

6 Future Development

Although the SIDE3 Loader currently implements a host of new features, by no means does this version represent the end of development. Indeed, the list of new features envisaged for future revisions is lengthy, and some ideas are listed below.

6.1 Loader

- In-place text editor to facilitate the editing of MAP files, shortcuts and other scripts
- Extension of the scripting facility provide further automation control
- A video player to complement the existing PDM player
- Broader use of DMA SD card transfers in order to speed up file copying, etc
- Utilisation of VBXE hardware 80 column text mode where available
- Settings and functionality relating to forthcoming hardware enhancements

6.2 FATFMS

- An enhanced version of DOS which uses the 'Shadow RAM' or extended memory to allow more functionality, buffers, FAT12 support, etc
- Improvements to the COPY command
- A system text editor
- A version of DOS which uses the OS SIO vector (relying therefore on the presence of U1MB)
- The ability to configure the system via INI files, etc
- Batch file processing

7 Feedback

If you find any bugs or have ideas for added functionality, please contact me by [email](#), or at AtariAge via PM (to user flashjazzcat) or in the SIDE3 Loader Discussion Thread.

Appendices

A References

The many sources of technical information which have proved invaluable while this software was in development include the following:

['Altirra Hardware Reference Manual' by Avery Lee](#)

['Mapping the Atari' by Ian Chadwick](#)

[SpartaDOS X Manuals and Documentation](#)

[Atari DOS 2.5 - 1050 Disk Drive Owner's Manual](#)

[SpartaDOS Construction Set Owner's Manual](#)

[Wikipedia: Master Boot Record](#)

[Wikipedia: The Design of the FAT File System](#)

['The Basics of the FAT File System' by Elm Chan](#)

[FatFs – Generic FAT File System Module by Elm Chan](#)

[MCP7951X/MCP7952X Battery-Backed SPI Real-Time Clock/Calendar](#)

[Microsoft Extensible Firmware Initiative FAT32 File System](#)

['How to Use MMC/SDC' by Elm Chan](#)

[SD Association: 'Simplified SD Card Specification'](#)

B Glossary

Terms used in this manual are defined below.

A8	Shorthand for 'Atari 8-bit computer'.
Address	A location in memory.
Altirra	A popular Atari 8-bit emulator for the Windows platform.
Append	Add to. When a file is open in 'append mode', information written is appended to the end of the file.
ASCII	The American Standard Code for Information Interchange.
ATASCII	Atari's variation on the ASCII character encoding scheme ('Atari-ASCII'). Incompatible with ASCII in several important respects, most notable in that the Atari EOL character (155/\$9B) is different to the ASCII CR (13/\$0D) and LF (10/\$0A) sequence.
ATR	Atari Disk Image file.
Bank	A block of memory occupying a specific address. SIDE3 has 1,024 8K banks of flash ROM and between 256 and 1,024 8K banks of SRAM. These banks may appear at \$8000-\$9FFF or \$A000-\$BFFF, allowing up to a 16KB window comprising any combination of SRAM and flash ROM, using a technique called 'bank switching'.
Bank Switching	Since the 6502 CPU uses 16-bit addressing, it can only address 64K of memory (65,536 bytes). Bank switching allows blocks of ROM and RAM in this address space to be swapped in and out, permitting the addressing of a much larger address space. SIDE3 uses bank switching to address up to 8MB of ROM and RAM, while the Atari 8-bit can access extended memory (where present) of up to 4MB capacity, usually via a banking window at \$4000-\$7FFF.
Batch file	A plain text file containing a list of commands to be executed one after the other.
BeweDOS	BeweDOS (or BW-DOS) is a SpartaDOS workalike which does not use RAM under the Atari operating system.
Binary	Base 2 numbering system, in which values are represented solely by binary digits of 0 or 1.
Bit	Binary digit.
Boot	The process of initialising the computer, either by turning it on or by executing a 'cold start'.
Buffer	Memory region used as temporary storage for data. Examples include type-ahead keyboard buffers and disk sector buffers.

Bus	A set of conductors carrying data and control signals within a computer system, to which pieces of equipment may be connected in parallel.
B, Byte	Binary number comprising 8 bits.
Checksum	A number which is the product of a mathematical function applied to a set of data, for the purpose of detecting transmission errors in the data.
CIO	Central Input/Output. The Atari uses the CIO for all communication with the keyboard, screen, and peripherals. SIDE3's 'FATFMS' DOS provides access to the hard disk via the CIO.
CLI	Command-line interface. See 'Command processor'.
Cold Start	Initialise the system as if the power was shut off and then re-applied without actually cycling the power switch. The SIDE3 loader triggers a cold start when – for example – booting a disk image.
Command	An instruction issued by the user to the computer.
Command Processor	The component of DOS which allows the user to interact with the system by typing commands at a prompt (commonly called the 'command prompt').
CPLD	Complex Programmable Logic Device. Many popular Atari 8-bit devices, including SIDE3, are based around CPLDs.
CPU	Central Processing Unit. This is the heart (or rather, the 'brain') of the computer reads and executes instructions in memory. The Atari 8-bit (usually) has a 6502C 'Sally' CPU running at ~1.7MHz.
CRC	Cyclic Redundancy Check. A two-byte number produced by performing a mathematical function on a set of data. Can be used as a checksum.
Current Directory	The directory implied if none is specified. The default 'current directory' is the main or 'root' directory. The current directory can be changed with the CHDIR command at the DOS prompt, or – in the SIDE3 loader – by opening a directory in the launcher menu (which causes that directory to become the current directory).
Cursor	Marker on the screen which indicates the location of the next operation.
Data	Information accessed or processed by software.
Decimal	Base 10 numbering system.
Default	The value or condition assumed if none is specified.
Density	The number of bytes per sector on a disk. Single Density (SD) and Enhanced Density (ED) disks have 128 bytes per sector, Double Density (DD) disks have 256 bytes per sector, and Quad Density (QD) or 'DD 512' disks have 512 bytes per sector. The SIDE3 loader deals exclusively with 'QD' or 'DD 512' disks, since the FAT file system can only be used on

storage volumes with 512-byte sectors. Disk images (ATRs), however, may comprise sectors of any of the three aforementioned sizes.

Device	An input/output component of the computer, whether internal (screen editor, keyboard) or external (disk drive, printer). The CIO provides identifiers for all the devices on the system ('D:' for the disk device, 'K:' for the keyboard device, etc).
Directory	A list (or index) of files on a disk storage volume. FAT, being a hierarchical file system, allows directories to exist inside of other directories. Such directories are called 'subdirectories.
Dispatch table	A table of pointers to functions or methods.
DOS	Disk Operating System. The software which manages communications between application software and the mass storage device.
DMA	Direct Memory Access. Any component of the computer which can directly access memory. SIDE3 includes a 'DMA engine' which facilitates the transfer of data from the SD card to memory (or memory to memory) in parallel to the Atari CPU (this is how large cartridge images are loaded into SRAM very quickly).
DRAM	Dynamic Random Access Memory. The Atari 8-bit – as manufactured – uses DRAM, whose contents must be periodically refreshed (this is done by the ANTIC chip). SIDE3 uses Static RAM (see 'SRAM).
Driver	Software which interfaces the computer with a specific piece of hardware. For example: FATFMS includes a driver which allows applications to interface with SD cards using standard CIO commands.
ECI	Enhanced Cartridge Interface. This is an extension to the Atari XE's cartridge connector which provides access to extra bus signals, including those found on the XL line's PBI connector.
FAT	File system developed by Microsoft and initially used by PC operating systems such as MS-DOS and early versions of Microsoft Windows. It is still widely used by USB storage devices, SD cards, and other portable storage devices.
FAT16	Version of the FAT file system which uses 16 bits per cluster number, allowing a maximum volume size of 2GB.
FAT32	Version of the FAT file system which used 32 (actually 28) bits per cluster number, allowing a maximum volume size of 32GB.
FATFMS	A new disk operating system (DOS) for the Atari 8-bit, initially released as a component of the SIDE3 loader. FATFMS currently uses the FAT16 and FAT32 file systems exclusively, thus providing direct file system compatibility between the Atari 8-bit and PC/Mac computers (via SD Card media).

FATFS	May refer to a file system driver which provides SpartaDOS X with (currently read-only) access to the FAT16 file system, or to Elm Chan's open-source FAT library. Not to be confused with FATFMS, which is a complete stand-alone disk operating system for FAT file systems.
File	A collection of information, usually stored as a named entity on a mass-storage device.
Filespec	The unique identifier of a file, comprising the file's name, its location in the directory structure, and the device on which the file is stored.
Firmware	Software permanently encoded in ROM or programmed to a microcontroller unit (see 'MCU') or CPLD.
Flash ROM	Non-volatile storage which can be electrically erased and reprogrammed. SIDE3 has 8MB of flash ROM.
Format	Initialise storage medium so that it can record information.
G, GB	In computing, G or GB refers to 1,024MB (or 1,073,741,824 bytes) or data or address space.
Handler	Essentially the same as a 'driver', and handler is software which provides access to a specific device. 'FATFMS' provides a disk handler and registers it in the CIO handler address table (see 'HATABS').
HATABS	The Atari CIO Handler Address Table. This table contains the device identifiers for all the handlers on the system and a pointer to the dispatch table for each handler. FATFMS adds the 'D' identifier to this table, and a pointer to the dispatch table in the disk handler.
Hard disk	A high-capacity storage device. Hard disks can be spinning platters or solid-state storage. In the context of SIDE3, the hard disk comprises one or more FAT-formatted partitions on a Secure Digital Card (see 'SD'). However, high-capacity disk images (ATRs) can also act as hard disks.
Hardware	The computer and its peripherals.
Header	Data at the beginning (head) of a file which provides information about the data contained in the file.
Hex	Abbreviation of 'hexadecimal', which is the base-16 numbering system, represented by the digits 0-9 and letters A-F. Hex values are commonly preceded by the dollar (\$) symbol, or by '0x'.
I/O	Input/Output between the computer and its devices and peripherals.
IDE	Integrated Drive Electronics is a standard commonly used by hard disks before the popularisation of SATA and PCI Express. Several hard disk host adapters for the Atari 8-bit use IDE hard disks. SIDE3 uses Secure Digital Input/Output (see 'SDIO') to communicate with the mass storage device, however.

Image	A binary representation of a storage volume or other media. Images are commonly contained in files, and may comprise a disk or cartridge ROM image.
IOCB	Input/Output Control Block. A 16-byte block of memory used to pass information to and from the CIO. There are eight IOCBs numbered 0-7 at address \$0340. IOCB 0 is usually used for the screen editor ('E:').
K, KB	In computing, K or KB refers to 1,024 bytes of data or address space.
M, MB	In computing, M or MB refers to 1,024KB (or 1,048,576 bytes) of data or address space.
Machine Code	Program code that the CPU can understand and execute.
MCU	Microcontroller Unit. A small computer on a single integrated circuit. Several popular Atari 8-bit peripherals are based around MCUs, since this allows the much more powerful hardware on the device itself to perform the majority of complex tasks (such as interfacing with the FAT file system using pre-written, open-source libraries).
Medium/Media	Non-volatile storage devices. SIDE3 uses Secure Digital (SD) cards (see 'SD').
MEMLO	The address of the first byte of usable memory above DOS. This value is stored at locations \$02E7/02E8 and represents the lowest load address for applications. When FATFMS is loaded, MEMLO is set to \$2000.
Memory Resident	A program which remains in memory after being loaded and remains there until the system is cold-started or powered off. DOS (FATFMS in the case of SIDE3) is memory resident.
Nibble	Four bits, or half a byte. A nibble can be represented by a single hexadecimal digit.
Non-Volatile	Storage media whose content is not lost when power is turned off. Examples are flash memory and ROM.
Optimise	Refers to the process of organising code or data (in memory or on storage media) for optimal execution or access.
OS	Operating System. The software which manages a computer's hardware and software.
Path	A list of subdirectory names describing the address of directory in which a file resides.
PBI	This can refer to the physical Parallel Bus Interface connector at the rear of the Atari 600XL and 800XL which allows external devices to access most of the important signals on the computer, or to the operating system protocols relating to PBI devices. The Ultimate 1MB (see 'U1MB'), for example, provides a 'PBI BIOS' which presents itself to the operating

system as if it were ROM on a device attached to the PBI connector.

Peripheral	In computing, a peripheral is a device attached externally to the computer. The SIDE3 cartridge is a peripheral.
Port	A connection point to the system; for example, the joystick port or the cartridge port.
Program	A set of instructions which cause the CPU to perform a specific set of operations.
Prompt	A signal that some user input is required. The FATFMS command processor prompts the user to enter commands.
RAM	Random Access Memory. Volatile storage (which loses its contents when the power is turned off) containing program code and data.
ROM	Read-Only Memory. Non-volatile storage (which maintains its contents when the power is turned off) containing program code and data. SIDE3 simulates cartridge ROM (when emulating cartridges) by loading data into SRAM and then making that SRAM read-only (so that it acts like ROM).
SD	Secure Digital. A proprietary, non-volatile flash memory card format developed by the SD Association (SDA) for use in portable devices.
Sector	The standard storage block used by floppy and hard disks. SIDE3 divides the hard disk (hosted on an SD Card) into 512-byte blocks. Disk images, however, may internally use 128, 256 or 512 byte sectors.
SIO	Serial Input/Output. The protocol used by the Atari for communication with devices attached to the serial bus. Parallel (PBI, ECI) devices may also be accessed via SIO once said devices have registered their handlers with the operating system. The SIDE3 loader communicates with the Ultimate 1MB PBI BIOS (where present) via the OS SIO entry point.
SIO2PC	One of the oldest Atari 8-bit peripherals which emulates disk drives by means of a PC. SIO2PC requires a program on the PC which presents disk images to the Atari (which believes it is interfacing with real disk drives).
Software	Program code and data which allow the computer to perform specific functions.
SpartaDOS	A powerful third-party disk operating system (from ICD) for the Atari 8-bit which supports subdirectories and hard disk volumes up to 16MB in size.
SDX	SpartaDOS X. Much more powerful cartridge-based successor to earlier SpartaDOS versions, produced by ICD. Later developed by FTe and currently maintained by DLT, who completely reverse-engineered the original ICD version and greatly enhanced it. SpartaDOS X is provided on the SIDE3 cartridge.
SPI	Serial Peripheral Interface. SIDE3 uses SPI Bus mode to communicate with

	SD Card media.
SRAM	Static RAM. Unlike DRAM, SRAM does not need to be periodically refreshed in order to maintain its contents. SIDE3 uses SRAM.
Subdirectory	Directories on disk which exist inside of other directories. Subdirectories are a useful tool for improving the organisation of files.
Syntax	The order and wording of commands or statements.
Truncated	Literally: shortened. Generally, this refers to a situation in which the amount of data received exceeded the available buffer space in size, causing part of the data to be discarded.
U1MB	Ultimate 1MB. Popular 1MB RAM upgrade and operating system switcher for the Atari 8-bit which has sold in large numbers, and whose functionality has gradually increased over time thanks to firmware updates. SIDE3 can work in tandem with U1MB where present.
VHD	Virtual Hard Disk. A Microsoft Windows disk image format.
Volatile	Storage whose content is lost when the power is turned off.
Warm Start	A system reset which does not obliterate the contents of system memory.
Wildcard	A symbol used as a substitute for one or more characters in a file or directory name. The SIDE3 Loader, FATFMS, and most disk operating systems support the '?' and '*' wildcard characters.
Word	A sixteen-bit integer comprising two bytes.
XIO	Extended Input/Output. CIO functions which fall outside of basic operations like Open, Close, Get and Put.

C Table of Figures

Figure 1: The browser/launcher menu	3
Figure 2: A scrolling long filename	4
Figure 3: Launcher context menu	7
Figure 4: The Device Pane	8
Figure 5: Partition Menu with context menu open	9
Figure 6: Partition Info	10
Figure 7: A FAT Partition open in the Launcher	10
Figure 8: The PDM Player	15
Figure 9: Changing the sort order	16
Figure 10: Searching for 'TUR'	17
Figure 11: APT in launcher with context menu open	19
Figure 12: The Mounts menu showing multiple mounted media	20
Figure 13: Performing a reverse lookup on a disk image	20
Figure 14: Tools menu	21
Figure 15: Changing the 'Sort by' setting	22
Figure 16: The second page of the Options menu	22
Figure 17: 'Add favourite' dialog	25
Figure 18: Editing a shortcut extender with alias extenders visible	26
Figure 19: The History folder	27
Figure 20: Setting up autorun on an executable	27
Figure 21: Selected items	28
Figure 22: The 'Get info' function applied to a folder	29
Figure 23: File copying in progress	30
Figure 24: Creating a new disk image (ATR)	31
Figure 25: Setting item properties	31
Figure 26: Assigning a drive number to an image	32
Figure 27: The command processor	37
Figure 28: Output of the DIR command	46
Figure 29: Output of the INFO command	48
Figure 30: Output of the VOL command	54

D Index

A

Altirra Emulator, 13, 58, 78, 79
 Append to a file, 79
 ATR disk image, i, iii, iv, 1, 2, 3, 6, 13, 14, 20, 21, 22, 23, 25, 26, 30, 33, 35, 36, 37, 38, 39, 42, 44, 50, 52, 56, 65, 79, 87

B

Bank switching, 79
 Batch file, 76, 79
 Boot, 3, 6, 26, 78, 80
 Buffer, 80
 BW-DOS, BeweDOS, 56, 70, 79

C

Central Input/Output System, 2, 13, 14, 25, 42, 44, 58, 66, 67, 68, 70, 71, 72, 80, 81, 82, 83, 86
 Cold Start, 80
 Command Line Interface, 1, 40, 75, 80
 Command Processor, ii, 13, 40, 80
 Context menu, i, 1, 6, 8, 9, 10, 11, 17, 20, 21, 22, 27, 28, 29, 30, 31, 32, 33, 35, 87
 CPU, 79, 80, 81, 83, 84
 Current working directory, iv, 19, 42, 55, 64, 67, 68, 80

D

Device, i, ii, 9, 10, 21, 38, 42, 49, 54, 70, 72, 73, 80, 81, 82, 83, 84, 87
 Directory, i, ii, iv, 17, 18, 38, 43, 48, 59, 63, 64, 67, 73, 80, 81
 Disk density, 13, 16, 81
 DMA, 1, 76, 81
 DOS, iii, iv, 2, 13, 14, 20, 25, 26, 35, 40, 42, 45, 52, 53, 55, 56, 57, 58, 59, 62, 70, 75, 76, 78, 79, 80, 81, 82, 84
 DRAM, 81, 85
 Driver, 56, 72, 75, 81, 82

F

FAT12, 57, 75, 76
 FAT16, 3, 4, 10, 11, 20, 50, 57, 59, 75, 82
 FAT32, 3, 10, 11, 20, 75, 78, 82
 FATFMS, iii, iv, 1, 13, 14, 20, 25, 26, 40, 41, 42, 43, 44, 56, 58, 59, 62, 70, 72, 75, 76, 80, 81, 82, 83, 84, 86
 File, i, ii, iii, iv, 1, 2, 4, 6, 9, 11, 12, 13, 14, 16, 17, 18, 19, 20, 21, 23, 25, 27, 28, 30, 31, 32, 33, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 65, 66, 67, 70, 71, 72, 73, 75, 76, 78, 79, 81, 82, 83, 84, 86, 87
 File types

ATR, i, iii, iv, 1, 2, 3, 6, 13, 14, 20, 21, 22, 23, 25, 26, 30, 33, 35, 36, 37, 38, 39, 42, 44, 50, 52, 56, 65, 79, 87
 COM, i, 13, 40, 42
 EXE, i, 13
 IMG, i, 13, 14, 21
 ISO, i, 13, 14, 21
 OBX, i, 13
 PDM, i, 1, 2, 13, 16, 17, 76, 87
 PDS, i, 13, 16
 VHD, i, 13, 14, 21, 86
 XEX, i, 1, 2, 6, 12, 13, 14, 20, 25
 Filespec, 49, 50, 82
 Flash ROM, 3, 79, 82

G

Get partition info, i, 11

H

Hard disk, 3, 13, 21, 36, 70, 80, 83, 85
 Hardware, ii, 2, 16, 26, 76, 78, 81, 83, 84
 HATABS, 82, 83

I

I/O, 83
 Image, ii, 14, 35, 36, 79, 83

J

Joystick input, i, 5, 6, 7, 8, 9, 10, 18, 25, 35, 84

L

Launcher menu, i, 1, 4, 8, 9, 11, 12, 18, 20, 21, 22, 28, 30, 37, 80, 87

M

MAP Files, i, 6, 13, 14, 36, 76
 MEMLO, 14, 45, 52, 84

N

Non-volatile memory, 24, 82, 84, 85

O

Operating System, 1, 14, 20, 39, 56, 70, 72, 76, 84, 85

P

Path, ii, iv, 11, 35, 36, 38, 40, 42, 44, 45, 46, 47, 48, 49,
50, 51, 52, 53, 54, 55, 56, 58, 60, 61, 63, 64, 65, 66,
67, 68, 73, 84

R

Rename FAT partition, i, 10

S

Shortcut files, i, 13, 17, 25, 27, 28
SIO, 3, 21, 70, 72, 73, 76, 85

SpartaDOS, 1, 2, 3, 6, 15, 20, 26, 37, 41, 42, 56, 58, 59,
60, 61, 62, 63, 64, 65, 66, 67, 70, 78, 79, 82, 85

SpartaDOS X, ii, 1, 2, 3, 6, 15, 20, 26, 37, 41, 62, 66, 70,
78, 82, 85

SRAM, 1, 26, 79, 81, 85

Subdirectory, 18, 41, 42, 44, 49, 53, 54, 59, 64, 65, 84, 85

Sub-tree, 31, 48

W

Wildcard, 38, 73, 86

