

# SPARTADOS



# X



FOR  
Atari  
400/800  
XL/XE  
only

## User Guide



DLT

Latest Edit: 8 August 2016

This manual was created using:

Linux Mint 17.3 Cinnamon 64 bit

Open Office Writer 4.1.2 for Linux

GIMP 2.8.10 for Linux

Fonts: Atari Classic Smooth, Khmer UI, Ubuntu Mono

We are proud to present the enhanced and remarkably upgraded

## **SpartaDOS X Version 4.48**

The basic idea of SpartaDOS emerged in the early 1980s: A powerful DOS resembling the concept of MS-DOS on the ATARI 8-bit platform. In late 1983 the concept came to live and developed into SpartaDOS 1.1, released in 1984. Version 2.x came to market in 1985. In 1986 ICD issued the last disk-based version: SpartaDOS 3.2x.

SpartaDOS X 4.17 was the first cartridge based version released in summer 1988. Last known commercial version 4.22 came 1995. A decade later this abandoned product got revived by its enthusiasts, who developed it to a new level and upgrade it frequently.

It matured to the 'SpartaDOS X Upgrade Project', aiming at the future development of the best disk operating system ever created for the Atari 8-bit computer. The project comprises the DOS itself, a toolkit, several add-ons, and the documentation. The new release again proves the power of SpartaDOS X, and its unique status in the Atari 8-bit world.

The SpartaDOS X package is configurable for every Atari 8-bit system. Please find more information and enhanced support by visiting the 'SpartaDOS X Upgrade Project' at <http://sdx.atari8.info/>.

See Appendix G for upgrades, updates and changes.

Take this guide for courtesy and keep Atari 8 bit computers alive!

Your DLT Team

### CREDITS

based on works done by	: ProfI, MMMG, DLT Ltd.
new code and design	: DLT Ltd.
emulator version	: dwhyte
hardware	: Pasiu/SSG, Jad/Phantasy, Zenon/Dial, DLT Ltd., Candle, MetalGuy66, Simius, lotharek, dropcheck, santosp, tf_hh
hosting	: krap.pl, Vasco/Tristesse
devtools	: DLT Ltd., Tebe/Madteam, others
guide	: Mikey, dely/BJB, DLT Ltd
other support	: ABBUC, Electron, Epi/Tristesse, FAiM, Guus Assmann, innuendo/DLT, jellonek, Jonathan "Flashjazzcat" Halliday, Kyle Dain, mono/Tristesse, Pin/Tristesse, Stephen J. Carden, Stryker, sup8pdct, Tomo Hornacek, FujiDude

As a quick reference the currently supported platforms:

intSDX128 and intSDX128 'flash'

Altirra and Atari800 emulators

IDE Plus 2.0 interface (contains MAN on-line help)

Ultimate 1MiB (contains MAN on-line help)

Incognito board (contains MAN on-line help, compatible with Ultimate1MB images)

SIDE & SIDE2 HDD cartridge (contain MAN on-line help)

SIC! Cartridge 128 KiB

SIC! Cartridge 256 KiB (contains MAN on-line help)

Turbo Freezer 2005 & 2011 (contain MAN on-line help, use with Freezer ROM only)

Maxflash 1 MiB

Maxflash 8 MiB (contains MAN on-line help)

Maxflash MyIDE+Flash

MyIDE II

Upgraded SpartaDOS X cartridge from ICD

SDX 128 'flash' cartridge

AtraX SDX 128 cartridge (use SDX2Atrax converter Add-on)

SpartaDOS X Super Cart

# **S P A R T A D O S   X**

---

**The Most Powerful 8-Bit  
Disk Operating System**

---

Original by ICD  
Enhanced Version by DLT Ltd.

## **Note - throughout this guide:**

SpartaDOS, SpartaDOS Construction Set, SpartaDOS X, SpartaDOS Toolkit, FlashBack!, Multi I/O, MIO, P: R: Connection, Printer Connection, UltraSpeed, US Doubler, MAC/65, BASIC XL, BASIC XE, OS/A+, and DOS XL were trademarks of FTe - all Abandonware.

Action!, also a former trademark of FTe is now freeware, released by Clinton Parker.

Atari 130XE, 65XE, 800XE, 800XL, 400/800, 810, 850, 1050, XF551, XE Game System, XEGS, XEP80, AtariDOS 2, AtariWriter, and AtariWriter Plus are trademarks of Atari Corp. - Abandonware.

ATR8000 is a trademark of Software Publishers, Inc. - Status unknown

Percom is a trademark of Percom Data Corp. - Out of business.

Axlon RAMPOWER is a trademark of Axlon, Inc - Abandonware.

MS-DOS is a trademark of Microsoft Corporation.

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Ltd.

ARC is a trademark of System Enhancement Associates - Abandonware.

Indus GT is a trademark of Indus Systems - Abandonware

Rana 1000 is a trademark of Rana Systems, Inc. - Abandonware.

MyDOS is Copyright © 1988 by WORDMARK Systems, Public Domain Software.

The Multiplexer is Copyright © 1990 by Computer Software Services.

BeweDOS is Copyright © 1995 by Jiří Bernášek - Freeware.

RealDOS is Copyright © 2011 by Stephen J. Carden - Shareware.

KMK/JŽ IDE V2 Plus is Copyright © 2011 by Konrad Kokoszkiwicz & Jacek Żuk.

Altirra -- Atari 8-bit home computer emulator is Copyright © 2008-2015 Avery Lee.

AspeQt up to V. 0.6.0 is Copyright © 2009 by Fatih Aygün,

Above V. 0.6.0 is Copyright © 2012 by Ray Ataergin.

RespeQt fork Copyright © 2015 by Joseph Zatarski and DrVenkman

SIO2BT is Copyright © 2014 by Marcin Sochacki

SpartaDOS X Programming Guide is Copyright © 2014 by DLT Ltd.

SpartaDOS X User Guide is Copyright © 2015-2016 by DLT Ltd.

SpartaDOS X - Published by ICD, Inc., 1988.

Last commercial version by FTe, 1995.

New Enhanced Version by DLT Ltd. 2008 - 2016

# Contents

## 1 Introduction

1.1 Synopsis .....	1
--------------------	---

## 2 An Introduction to SpartaDOS X

2.1 What is DOS? .....	3
2.2 SpartaDOS X .....	3
2.3 The Command Processor .....	3
2.4 Getting Started .....	3
2.5 Formatting a Disk .....	4
2.6 Disk Directory .....	5
2.7 Creating Test Files .....	6
2.8 Setting the Time and Date .....	7
2.9 Parameters .....	7
2.10 Copying Files .....	8
2.11 Erasing Files .....	8
2.12 Wildcards .....	8
2.13 Directories .....	10
2.14 The Current Directory .....	11
2.15 Running Programs .....	12
2.16 BASIC, CAR, and X .....	12
2.17 Building Batch Files .....	13
2.18 Practice .....	14
2.19 DOS 2 Equivalents .....	14

## 3 SpartaDOS X Overview

3.1 Drives .....	15
3.2 Filenames .....	15
3.3 Filename Extensions .....	15
3.4 Wild Card Characters .....	16
3.5 File Attributes .....	16
3.6 Time/Date Stamp .....	17
3.7 Directories .....	17
3.8 Pathnames .....	17
3.9 Command Length .....	18
3.10 Filespec .....	18
3.11 Device Identifiers .....	18
3.12 Default Drive and Directory .....	20
3.13 Volume Names .....	21
3.14 Disk Format Compatibility .....	21
3.15 Using External Cartridges with SDX .....	22
3.16 Drivers .....	22
3.17 Memory Management .....	23

## 4 The Command Processor - Commands

4.1 APPEND - add destinations to system path .....	27
4.2 ARC - archive files .....	28
4.3 ATR - file attributes .....	32
4.4 BASIC - enter internal BASIC .....	33
4.5 BLOAD - binary load .....	35

4.6	BOOT - start a program at system start up .....	36
4.7	CAR - enter a cartridge .....	37
4.8	CHDIR - change directory .....	40
4.9	CHKDSK - check disk .....	41
4.10	CHTD - change time/date stamp .....	42
4.11	CHVOL - change volume name .....	43
4.12	CLR - clear variables .....	44
4.13	CLS - clear screen .....	44
4.14	COLD - cold start .....	45
4.15	COMMAND - The Command Processor .....	46
4.16	COMP - compare files .....	47
4.17	CON - console modes .....	48
4.18	COPY - copy files .....	49
4.19	DATE - show and set date .....	54
4.20	DELTREE - delete directory tree .....	56
4.21	DEV - kernel devices .....	57
4.22	DF - disk free .....	58
4.23	DIR - directory & DIRS - short directory .....	59
4.24	DUMP - display file contents .....	61
4.25	ECHO - echo command line input .....	62
4.26	ED - editor .....	63
4.27	ERASE - erase files .....	65
4.28	FIND - find files .....	66
4.29	FMT - format text .....	67
4.30	FORMAT- format disk .....	68
4.31	KEY - keyboard buffer .....	72
4.32	LESS - view text .....	73
4.33	LOAD - un-/load a file .....	75
4.34	MAN - display online documents .....	76
4.35	MAP - map drives .....	79
4.36	MDUMP - display memory contents .....	80
4.37	MEM - display memory configuration .....	81
4.38	MENU - menu program .....	84
4.39	MKDIR - make directory .....	88
4.40	MORE - text viewer .....	89
4.41	PATH - set search directory .....	90
4.42	PAUSE - a pause .....	91
4.43	PEEK - display memory location content .....	92
4.44	POKE - change memory location content .....	93
4.45	PROMPT - set system prompt .....	94
4.46	PWD - print working directories .....	96
4.47	RENAME - rename files .....	97
4.48	RENDIR - rename directory .....	98
4.49	RMDIR - remove directory .....	99
4.50	RS232 - load RS232 driver .....	100
4.51	SAVE - save binary data .....	101
4.52	SET - set environment variable .....	102
4.53	SETPATHS - set current path .....	103
4.54	SIOSET - SIO speed control .....	104
4.55	SORTDIR - sort filenames .....	106
4.56	SWAP - swap drives .....	107
4.57	TD - time/date display .....	108



4.58	TIME - show and set time .....	109
4.59	TYPE - display file contents .....	110
4.60	UNERASE - restore erased files .....	111
4.61	VER - SDX version information .....	112
4.62	VERIFY - write verify .....	112
4.63	X - execute without cartridge .....	113
<b>5</b>	<b>The Command Processor - Advanced Features</b>	
5.1	Running Programs .....	115
5.2	Batch Files .....	115
5.2.1	Default Batch File .....	116
5.2.2	Conditionals .....	116
5.2.2.1	IF EXISTS Conditional .....	116
5.2.2.2	IF ERROR Conditional .....	117
5.2.2.3	IF INKEY Conditional .....	117
5.2.2.4	IF FILE conditional .....	118
5.2.3	Existence of environment variables .....	118
5.2.4	Comparisons .....	118
5.2.5	Loops .....	118
5.2.6	GOTO Jumps .....	119
5.2.7	INKEY Command .....	120
5.2.8	Procedures .....	120
5.2.9	Other Batch File Commands .....	121
5.3	I/O Redirection .....	121
5.4	Pipes .....	122
5.5	Search Path .....	122
5.6	Automatic evaluation of environment variables .....	123
<b>6</b>	<b>Programming with SpartaDOS X</b>	
6.1	SDX Functions from BASIC .....	125
6.2	Notes on the Default Drive .....	125
6.3	Accessing the 'Kernel' Through CIO .....	125
6.3.1	Open File .....	126
6.3.1.1	Directory formatting attributes .....	127
6.3.1.2	Accessing the Raw Directory .....	128
6.3.1.3	Scan Mode .....	128
6.3.2	Rename File(s) (RENAME) .....	129
6.3.3	Erase File(s) (ERASE) .....	129
6.3.4	Protect File(s) (ATR +P) .....	129
6.3.5	Unprotect File(s) (ATR -P) .....	130
6.3.6	Set File Position - POINT .....	130
6.3.7	Get Current File Position - NOTE .....	131
6.3.8	Get File Length .....	132
6.3.9	Load a Binary File (LOAD) .....	132
6.3.10	Change Default Drive & Directory .....	132
6.3.11	Create a Directory (MKDIR) .....	133
6.3.12	Delete a Directory (RMDIR) .....	133
6.3.13	Change Current Directory (CHDIR) .....	133
6.3.14	Set Boot File (BOOT) .....	134
6.3.15	Set Attributes (ATR) .....	134
6.3.16	Format a Disk (FORMAT) .....	135
6.3.17	Get Disk Information (CHKDSK) .....	135

6.3.18	Get Current Directory Path (CHDIR) .....	137
6.4	SpartaDOS User Accessible Data Table .....	139
6.5	Decoding the drive identifier .....	142
6.6	Symbols .....	143
6.7	Vectors Under the OS ROM .....	143
6.8	Page Seven 'Kernel' Values .....	145
6.9	Using the CON: Drivers in Own Programs .....	147
6.9.1	Detecting the extension .....	147
6.9.2	Enabling and disabling the extended mode .....	148
6.9.3	The 'direct write' entry point .....	148
6.9.4	Scrolling the display up .....	149
6.9.5	Other functions .....	149
<b>7</b>	<b>Technical Information</b>	
7.1	SpartaDOS File System .....	151
7.1.1	Boot Sectors .....	151
7.1.2	Bit Maps .....	154
7.1.3	Sector Maps .....	154
7.2	Directory Structure .....	154
7.2.1	Exploring Disks .....	155
7.2.2	PERCOM Extensions .....	155
7.3	Direct Disk Access .....	156
<b>8</b>	<b>Configuring Your System</b>	
8.1	SDX Boot Process .....	157
8.1.1	Boot Drive Number .....	157
8.1.2	Memory Configuration .....	157
8.2	CONFIG.SYS File .....	157
8.2.1	USE Command .....	158
8.2.2	SET Command .....	159
8.2.3	DEVICE Command .....	159
8.2.4	ECHO Command .....	159
8.2.5	MERGE Command .....	159
8.2.5.1	Basic usage .....	159
8.2.5.2	Advanced use .....	160
8.2.5.3	Special usage .....	160
8.3	Config Selector .....	161
8.4	Character Sets .....	161
8.5	Important system variables .....	161
8.6	Drivers .....	165
8.6.1	File System Drivers .....	165
8.6.1.1	SPARTA.SYS - SDFS disk format .....	165
8.6.1.2	ATARIDOS.SYS - AtariDOS 2.0 file system .....	166
8.6.2	Block I/O Drivers .....	167
8.6.2.1	SIO.SYS - serial and parallel I/O .....	167
8.6.2.2	INDUS.SYS - Indus high speed SIO .....	168
8.6.2.3	RAMDISK.SYS - SDX ramdisk .....	168
8.6.2.4	PBI.SYS - special parallel I/O driver .....	170
8.6.3	Timekeeping Drivers .....	170
8.6.3.1	ARCLOCK.SYS - Atari Real Clock .....	170
8.6.3.2	IDEPTIME.SYS - IDE 2.0+ RTC .....	171
8.6.3.3	ULTIME.SYS - Ultimate, SIDE 1&2, SuperCart RTC .....	171

8.6.3.4	JIFFY.SYS - Atari software clock .....	172
8.6.3.5	RTIME8.SYS - R-Time 8 RTC .....	172
8.6.3.6	Z.SYS - access to SDX timekeeping functions .....	172
8.6.4	Screen Drivers .....	174
8.6.4.1	XEP80.SYS -Atari XEP 80 column display .....	174
8.6.4.2	QUICKED.SYS - accelerated screen output .....	175
8.6.4.3	CON64.SYS - 64-column display .....	175
8.6.5	Application Drivers .....	177
8.6.5.1	COMEXE.SYS - cartridge management .....	177
8.6.5.2	RUNEXT.SYS - file association .....	177
8.6.6	Other Drivers .....	180
8.6.6.1	DOSKEY.SYS - extended command line .....	180
8.6.6.2	ENV.SYS - environment extension .....	182
<b>9</b>	<b>The SpartaDOS X Toolkit</b>	
9.1	Pre-Configured Batch Files .....	183
9.2	Boot Loader .....	184
9.2.1	SDLOAD - A loader for games and demos. ....	184
9.2.2	INIDOS.SYS - restart a deactivated SDX .....	185
9.3	Compatibility .....	185
9.3.1	VPRINT.SYS – the respective call known from SD 3.2 .....	185
9.4	Dynamic Link Libraries .....	186
9.4.1	UNIXTIME – link library for assembler programs .....	186
9.4.2	FSEL - file selector library for assembler programs .....	186
9.5	Drivers .....	187
9.5.1	PCLINK.SYS - link to a PC file server .....	187
9.5.2	Screen Drivers .....	188
9.5.2.1	S_VBXE.SYS - use the Video board XE .....	188
9.5.2.2	RC_VBXE.SYS - VBXE driver using less memory .....	189
9.5.2.3	VBXE.SYS – minimalistic driver for VBXE .....	189
9.5.2.4	RC_GR8.SYS - 80-column basis .....	189
9.5.2.5	CON.SYS - 80-column console .....	190
9.5.3	Drivers for Turbo Boards .....	191
9.5.3.1	Rapidus Turbo Board MMU driver .....	191
9.5.3.2	Rapidus Turbo Board – RAM disk driver .....	192
9.5.3.3	65C816 RAM Speed Test .....	192
9.5.3.4	65C816 RAM Integrity Test .....	193
9.5.3.5	65C816 Interrupt Driver .....	193
9.5.4	Other Drivers .....	194
9.5.4.1	CAD.SYS - terminate any software by keypress .....	194
9.5.4.2	CPMFS.SYS - read Indus CP/M disks .....	195
9.5.4.3	USER - change the user on a CP/M disk .....	195
9.5.4.4	FATxxx.SYS - read MS-DOS filesystem .....	196
9.5.4.5	CHKFAT – verify FAT compability .....	197
9.5.4.6	PBI.SYS .....	197
9.5.4.7	KTRACE – trace SDX kernel calls .....	198
9.5.4.8	RAMD816L.SYS - SDX ramdisk for 65C816 systems .....	199
9.6	Disk Tools .....	200
9.6.1	CleanUp X - The SDX Disk Fixer .....	200
9.6.2	Eddy - advanced sector editor .....	201
9.7	TSR Programs for SDX 4.2x .....	201

9.8	Shells .....	202
9.8.1	MyDUP .....	202
9.8.2	Sparta Commander .....	202
9.9	Utilities .....	203
9.9.1	APETIME - get time & date from APE .....	203
9.9.2	CDD - change drive and directory .....	204
9.9.3	CRC32 - compute a cyclic redundancy check .....	204
9.9.4	DD - derive data .....	205
9.9.5	DELDUP - delete duplicates .....	206
9.9.6	DPOKE - double poke .....	207
9.9.7	DU - disk used .....	207
9.9.8	FA_REG - Fast Assembler - registered version .....	208
9.9.9	FSTRUCT - display file structure .....	208
9.9.10	HDSC - hard disk sector copier .....	208
9.9.11	HEXDEC - hex-dec converter .....	209
9.9.12	INVERSE - invert input .....	210
9.9.13	LS - list files .....	210
9.9.14	MACH - machine details .....	211
9.9.15	MD5 - hash tool .....	211
9.9.16	MEMINFO - memory information .....	212
9.9.17	MKATR - make an atr image .....	213
9.9.18	NVWGL & NVPEEK - inspect and manipulate NV-RAM .....	214
9.9.19	PHD - push directory .....	214
9.9.20	PLD - pull directory .....	215
9.9.21	RDDUMP & RDLOAD - ramdisk save & restore .....	215
9.9.22	RPM - check rotation speed .....	216
9.9.23	RUN - run the code .....	217
9.9.24	SL - symbol list manipulation .....	218
9.9.25	STAT - status information .....	219
9.9.26	TAR - archiver for SD .....	220
9.9.27	TREE - display directory tree .....	222
9.9.28	TSR.SYS - enhance the number of TSR programs .....	222
9.9.29	WC - words counter .....	223
9.9.30	XFCONF - density selection for floppy disk drives .....	223
9.9.31	XT - execution time .....	224
9.9.32	XVER - SDX build version .....	225
9.10	Man Pages .....	225

## 10 SpartaDOS X Add-Ons

10.1	By DLT .....	227
10.1.1	SDX Imager .....	227
10.1.2	SDX2Atrax converter .....	227
10.1.3	MUXTIME for SDX .....	227
10.1.4	Emu-pack .....	227
10.1.5	S2I - a SIO2IDE-specific tool .....	227
10.1.6	S2S - a SIO2SD-specific tool .....	227
10.1.7	MNT - a KMK/JŽ IDE 1.0 / IDEa specific tool .....	227
10.2	Contributed by Bober .....	228
10.2.1	Aprplr - APR animation player .....	228
10.2.2	Core Wars - SDX Core Wars package .....	228
10.2.3	Toys - 4 small programs .....	228

10.3	Contributed by FujiDude .....	228
10.3.1	AAC - ASCII<->ATASCII converter .....	228
10.4	Contributed by mono .....	228
10.4.1	Worm - a screen saver .....	228
10.4.2	Drum Machine - convert your Atari into a drum kit! .....	228
10.4.3	CMC Play - Chaos Music Composer player .....	228
10.4.4	FC Play - Future Composer player. ....	228
10.4.5	MPT Play - Music Protracker player. ....	228
10.4.6	RMT Play - Raster Music Tracker player. ....	228
10.4.7	ST Play - Sound Tracker player. ....	228
10.4.8	TMC Play - Theta Music Composer player. ....	229
10.5	Diamond Desktop .....	229

## Appendices

### A DOS limitations

The maximum capabilities list .....	231
-------------------------------------	-----

### B Error Messages

B.1 Description of all error messages .....	233
B.2 Error Message Summary .....	238

### C Command Summary - Alphabetical

Quick Reference List .....	239
----------------------------	-----

### D Command Summary - By Function

D.1 Batch Files .....	245
D.2 Configuration Commands .....	246
D.3 Directory Commands .....	246
D.4 Disk Maintenance Commands .....	247
D.5 File Maintenance Commands .....	247
D.6 Running Programs .....	248
D.7 Command Processor Options .....	248
D.8 Time - Date Support .....	249
D.9 Utilities And Programming Aids .....	249

### E Miscellaneous Notes

E.1 Using Turbo-BASIC XL with SDX .....	251
E.1.1 Hardware Configuration .....	251
E.1.2 System Configuration .....	251
E.2 Using AUTORUN.SYS files .....	252
E.2.1 Applications .....	252
E.2.2 Handlers .....	252
E.2.3 BASIC Program Loaders .....	252
E.2.4 Using Batch Files .....	252
E.3 Using ACTION! with SDX .....	252
E.4 Using BASIC XL/XE with SDX .....	253
E.5 Using BASIC XE Extensions .....	253
E.5.1 Loading the Extensions .....	253
E.5.2 Other Issues .....	253
E.6 Using MAC/65 and DDT with SDX .....	253
E.7 Using AtariWriter Plus with SDX .....	254

**F System Drivers Summary**  
 Alphabetical quick reference ..... 255

**G Changes, Innovations And Way Ahead**  
 G.1 Kernel ..... 257  
 G.2 Drivers and resident programs ..... 257  
 G.3 Library ..... 258  
 G.4 Formatter ..... 258  
 G.5 Utilities ..... 259  
 G.6 SDX Toolkit ..... 260  
 G.7 To-Do List ..... 262

**H Glossary**

**Table of Figures**  
 Screenshots ..... 279

**Index**  
 In alphabetical order ..... 281

# 1 Introduction

## 1.1 Synopsis

Congratulations for your choice to use SpartaDOS X. We feel confident that, after you become familiar with, you will find it to be the most powerful disk operating system ever produced for your Atari 8-bit computer.

Throughout this guide some specific abbreviations are used to ease the wording:

- SD** is for SpartaDOS; comprises legacy versions and/or a general information.
- SDTK** notes the legacy SpartaDOS Toolkit, developed by ICD.
- SDFS** is the SpartaDOS File System; the file system SD uses when doing read or write operations to media like formatting, loading, saving.
- SDX** stands for SpartaDOS X; what this guide is all about.
- SDXTK** stands for SpartaDOS X Toolkit; descriptions included in this guide as well.

More special terms, e. g. A8, may be found in the glossary<sup>1</sup>.

**Chapter 2** illustrates SDX with step-by-step examples for novices who meet any version of SD for the first time. A guide to learn how SDX operates and how you can take advantage of its power.

**Chapter 3** describes the general operation of SDX and the ways how things are being handled when using the command processor.

**Chapter 4** explains the SDX command set in detail, providing complete information for each command. Until you become familiar with this command set, you will probably be referring to this chapter often.

**Chapter 5** discusses advanced features such as batch files, I/O redirection and search paths. This information is not required to use SDX, but can help you take full advantage of these features to save time, perform complex tasks easily, and configure your system for optimum use.

**Chapter 6** covers programming, showing BASIC statements for novice programmers and some machine language access to the inner workings of SDX for advanced programmers. Examples are given in BASIC and assembly language to aid the programmer in integrating these concepts into the own programs. Additionally, a programming guide is now available.

**Chapter 7** focuses on a technical and detailed look at the medium and directory structure of SDX. This chapter will probably not be of interest to many users but was included for those interested in creating complex programs.

---

<sup>1</sup> Appendix H - Glossary.

**Chapter 8** provides information for tailoring the configuration of SDX to your system. In this way you can take full advantage of your special hardware setup to gain maximum flexibility and control.

**Chapter 9** adds the information about the contents of the SDXTK. There are quite some drivers, tools, utilities, batch files and man pages found, which enhance the power of SDX even further.

**Chapter 10** tells you about special add-ons for the SDX power user.

**Appendices** cover such topics as error messages, command summaries, common problems and solutions, new developments, and hints for the use of special software as programming languages, information about the latest changes and bug-fixes for the current version.

SDX is by far the most complex disk operating system available for the A8. The very things that make SDX so powerful may also make it more difficult to learn than other disk operating systems. If you should have any problems at any time with SDX or any other former ICD A8 product, we advise you to post respective questions on supporting A8 fora on the Internet. You will find all information about the 'SDX Upgrade Project' on its homepage at

<http://sdx.atari8.info/>

**A note about incompatibilities:** There are some programs that just will not work with SDX (or any other DOS for that matter). Some programs are protected or have a DOS built into them. SDX is more compatible with other programs than any previous version of SD, but there will always be a handful of programs that will not work.

Also a patched or modified OS may hamper the use of SDX on your machine. If you experience problems try an unmodified Atari OS first.

**Technical Advice:** The full potential of SDX is available only on XL/XE computers. However, it can also be used with Atari 400/800 machines having at minimum 48 KiB<sup>2</sup> of RAM considering the differences.

We strongly recommend to use a machine having at minimum 128 KiB of memory. A well equipped system should have at least 320 KiB of overall memory.

**Note:** In general this guide applies to all provided platform specific versions of SDX. Please refer to the respective platform manual first, if you should encounter problems.



## 2 An Introduction to SpartaDOS X

This chapter is for those of you without prior experience with any version of SD. As a user of legacy versions of SD you may wish to simply review this chapter and move on to chapter 3, which outlines the new features found in SDX version 4.4x by DLT.

### 2.1 What is DOS?

DOS stands for Disk Operating System. The primary purpose of DOS is to allow the computer to communicate with one or more disk drives. In practice, a DOS provides many more useful features and complements the Atari OS to a full functioning system.

### 2.2 SpartaDOS X

SDX serves these purposes and more. It may take a little longer to learn than other types of DOS for the A8 computer, but it will provide such a degree of power and control over the computer that the learning period will be well worth it. This chapter contains several examples of elementary SDX operations in a tutorial format. Please read through it and perform the examples. Do take time to experiment. This will get you well on the way to becoming a SDX power user.

### 2.3 The Command Processor

SDX differs from AtariDOS types in many ways, most apparently by the user interface, the way in which you communicate with DOS and DOS communicates with you. Most other DOS are menu driven: all options available are displayed on the screen and may be selected with a single key stroke. SDX uses a command processor (CP): at a prompt, the full command is typed. Those of you familiar with any PC DOS, CP/M, UNIX, and OSs on other computers will recognize the CP interface. Each of these user interfaces have strong and weak points, but when you become familiar with the operation of any SD you will find its CP interface to be very powerful and flexible. A menu is also included to make file maintenance more easy.

### 2.4 Getting Started

To get started, insert the SDX cartridge into the (left) cartridge slot (on an Atari 800) having at least 48 KiB of memory and turn the computer on. If you happen to use a different host than a cartridge for your SDX, please attach it to your system as explained in the respective manual and turn it on. SDX will check the system and display some information depending on the hardware configuration of your A8 computer, e. g. like this:

```
R-Time 8 not present
Ramdisk not installed
No extended RAM

SpartaDOS X 4.47a 2-11-2013
Copyright (C) 2013 by FTE & DLT

D1:■
```

Fig. 1: Boot Information - 48 Or 64 KiB Of Memory

It indicates a 400 or 800 with 48 KiB or any XL/XE computer with 64 KiB of overall RAM, and it tells you about the memory found by SDX: 48 KiB of main memory. The message about the real time clock 'R-Time 8' is of no importance yet.

If your XL/XE computer comes with 128 KiB of RAM, you will see this:

```

R-Time 8 not present
Ramdisk not installed
64k left for XE programs

      SpartaDOS X 4.47a 2-11-2013
      Copyright (C) 2013 by FTE & DLT

D1:■
  
```

Fig. 2: Boot Information - 128 KiB Memory

If there are more than 128 KiB of RAM on XL/XE (Port B type) or more than 48 KiB on 400/800 (Axlon type) detected, only the real time clock and version information is displayed. This notes that SDX has found more than sufficient memory in your Atari to create a convenient environment.

```

R-Time 8 not present

      SpartaDOS X 4.47a 2-11-2013
      Copyright (C) 2013 by FTE & DLT

D1:■
  
```

Fig. 3: Boot Information - Sufficient Memory Found

SDX has an intelligent memory management, which tests and configures the memory during boot up<sup>3</sup>. The default setup uses memory found according to these schemes:

#### 400/800

- 48 KiB main memory,
- 16 KiB extended memory for SDX internal use (if extended RAM is available),
- the rest of the extended memory is used as ramdisk 'O':.

#### XL/XE

- 48 KiB main memory,
- 14 KiB memory under ROM,
- 64 KiB memory reserved for BASIC XE programs (with 64 KiB extended RAM),
- 16 KiB extended memory for SDX internal use (with more than 64 KiB ext. RAM),
- the rest of the extended memory is used as ramdisk 'O':.

Type **CHKDSK O: /X** and press the <RETURN> key to see detailed information about your ramdisk, if your A8 has sufficient extended memory.

**Notes:** Always press <RETURN> after input typed. To restart SDX type **COLD** and press <RETURN>. There is no need to switch your Atari off and on again and again during this introduction except it might lock up.

## 2.5 Formatting a Disk

Before you start to explore the command processor, you need to have a floppy disk with which to experiment, or use any other medium, if you are familiar with it. Place a new disk in drive #1, type **FORMAT** (→ and press <RETURN>). You do not need to type the 'D1:'. Throughout this chapter, you will only need to type the text after the prompt.

<sup>3</sup> See Chapter 8 for more detailed information.

The 'SpartaDOS Formatter' menu shows information depending on the drive hardware found. The formatter will detect the selected drive, partition or drive emulation, given it has been setup as an A8 compliant drive with the respective drive software.

The blinking cursor in the unit field asks for input of the drive id. Press the <1> key to select drive #1 or any other valid drive number depending on your setup. Next the tool will present the facts it found about the drive like number of sectors, sector size and number of bytes. Those values will be used to format the selected drive. Depending on the drive information detected the formatter may see it as a hard drive partition, displaying in the bottom line 'Format n/a' (not available). In this case please use the option 'Build Directory' and follow the directions prompted on the screen. More about the 'FORMAT command' is to be found in chapter 4.

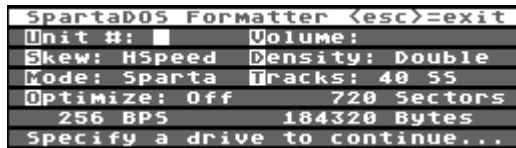


Fig. 4: Formatter Menu

Now, press the <V> key and type **TESTDISK** (→ and press <RETURN>). This sets the volume name of the disk to 'TESTDISK'. Make sure you have a new disk in drive #1, press <F> and <RETURN> to format the disk. Follow the prompted information and finish to format your new 'testdisk'. The other options on this menu are fully described in chapter 4 under the FORMAT command. You may just ignore them for this exercise.

The drive id in the formatter menu is reflected by capital letters A to O. Having pressed 1 you will see an 'A' with 'Unit #:'. You may use the letters instead, which is mandatory for drive numbers higher than 9.

You may also format the ramdisk of your computer using the formatter. The standard drive id for it is 'O'. Experiment a bit, if you like.

When finished with formatting press <ESC> to leave the formatter menu and go back to the system prompt.

## 2.6 Disk Directory

Now type **DIR**. You will be prompted with the information about the just formatted medium. You may type **CHKDSK A: /X** to see more details about the medium in drive #1 (or A).



Fig. 5: First Formatted Disk

## 2.7 Creating Test Files

The disk is empty now. To be able to experiment with commands, you need some files on it. With a XL or XE computer (except 1200XL) type **BASIC**.

With a 400, 800 or 1200XL make sure you have the Atari BASIC cartridge installed in the top of the SDX cartridge. Otherwise, turn the computer off, insert the BASIC cartridge into the SDX cartridge, and turn the computer on again. With a 400/800, it is necessary to 'fool' the computer into thinking the cartridge door is closed by holding the door switch down. This can be accomplished by sticking a small item into the hole located on the front right edge of the cartridge area. Instead of BASIC type **CAR**.

If BASIC is available in one or the other way, regardless of the computer type, you should now see the BASIC prompt **READY**. Next type in the short BASIC program as depicted on the screenshot, store it (**SAVE**) on disk and execute it (**RUN**).

```

D1: BASIC
READY
10 OPEN #1,8,0,"D1:TEST.DAT"
20 FOR X=0 TO 255
30 PUT #1,X
40 NEXT X
50 CLOSE #1
60 END

SAVE "D1:TEST.BAS"

READY
RUN
READY
█

```

Fig. 6: Create New Files On Test Disk

Now type **DOS** to get back to the DOS prompt on the command line interface.

Next type **DIR** and you will see:

```

SAVE "D1:TEST.BAS"
READY
RUN
READY
DOS
D1: DIR
Volume: TESTDISK
Directory: MAIN
TEST      BAS      153   5-11-11  14:49
TEST      DAT      256   5-11-11  14:49
710 FREE SECTORS
D1: █

```

Fig. 7: New Files On Test Disk

The directory format of SDX is different from any other DOS. The first eight characters (four in this case) are the filename. The next three are the extension. The first number to the right is the length of the file in bytes, not sectors. This is followed by the date and

time. Your date, time, and free sectors count may vary. 'TEST.BAS' is the BASIC program you just typed in, and 'TEST.DAT' is the data file it created. This directory format is standard for the SDFS, which holds more information than any other file system on the A8 platform.

## 2.8 Setting the Time and Date

If you do not have a supported hardware clock, SDX will by default install a software clock, which needs to be set manually. This feature allows you to keep track of when your files were created and which is the latest version of a particular file. Following the next steps will get you to the information shown on this screen shot:

```

SpartaDOS X 4.47 | Sat 16-Nov-13 | 21:02:43
D2:DATE
Current date is: 25-01-06
Enter new (DD-MM-YY): 16-11-13

D2:TIME
Current time is: 20:17:06
Enter new (HH:MM:SS): 21:01

D2:CHTD TEST.BAS
Changing TD in TEST.BAS

D2:DIR
Volume:
Directory: MAIN
TEST   BAS      153 16-11-13 21:01
TEST   DAT      256 25-01-06 20:16
      710 FREE SECTORS

D2:TD ON
D2:█

```

Fig. 8: Time And Date Display Line

Type **DATE**. The current system date is shown and you will be prompted to put in the actual date. Enter the date format shown in brackets on your screen where 'DD' is the month, 'MM' is the day, and 'YY' is the year.

Next type **TIME**. Again, the actual time and date displayed may vary from the examples. Now enter the new time. The time should be in the format 'HH:MM:SS' and is based on a 24 hour clock (for example, 5:30 PM would be 17:30:00). Now time and date are set.

Type **CHTD TEST.BAS**. This will **CH**ange the **T**ime and **D**ate of 'TEST.BAS' to the current time and date. Now type **DIR**. You will see that the file 'TEST.BAS' now really has the current time and date.

Next type **TD ON** to turn on the **T**ime and **D**ate display at the screen top

The TD display line may be disabled by typing **TD OFF**.

If you will use the software clock of SDX, it is helpful to adjust the file AUTOEXEC.BAT on your boot drive by incorporating the commands TIME and DATE. An example is shown at the end of this chapter. See chapter 5 for more details about batch files.

## 2.9 Parameters

Many commands require parameters. In the example for the command CHTD, the filename part 'TEST.BAS' is a parameter. With TD, 'ON' and 'OFF' are parameters. A parameter is an additional information passed to the command by typing it after the

command on the same line. Many commands use more than one parameter. Parameters should be separated from the command and from each other by spaces (although commas are allowed with SDX and certain commands). Some commands, such as TIME and DATE, use no parameters. Some parameters are required, while others are optional. Often default values are assumed if no parameters are provided. Since this information varies from command to command, consult chapter 4 for the various required and optional parameters of each command.

## 2.10 Copying Files

The command to copy files is COPY. This can be used to copy a file from one disk or directory to another or to copy a file to the same disk with another name or path.

Type **COPY TEST.BAS MAKEDAT.BAS** and list the directory (with the DIR command). Note that the file 'MAKEDAT.BAS' has the same length, time, and date as the file 'TEST.BAS', because it is just another copy of the same file. To copy a file from one disk to another with only one drive you may use a ramdisk if available. Otherwise you must use the MENU program<sup>4</sup>.

## 2.11 Erasing Files

Erasing a file removes it from the disk. While it is possible in some cases to recover a file that has been erased, it is a good idea to be very careful when erasing files.

Type **ERASE TEST.BAS** and list the directory. The file TEST.BAS is gone.

```

D1:DIR
Volume:   TESTDISK
Directory: MAIN

TEST     BAS      153  15-02-11  15:07
TEST     DAT      256  25-01-06  20:15
MAKEDAT  BAS      153  15-02-11  15:07
      708 FREE SECTORS

D1:ERASE TEST.BAS

D1:DIR
Volume:   TESTDISK
Directory: MAIN

TEST     DAT      256  25-01-06  20:15
MAKEDAT  BAS      153  15-02-11  15:07
      710 FREE SECTORS

D1:█

```

Fig. 9: Erase Files

## 2.12 Wildcards

Most SDX commands allow you to select more than one file by using wildcards in place of one or more characters. In poker, a wild card can be substituted for any other card. Wildcards in all versions of SD perform a similar function.

There are two wildcards used in SD as in most other DOS types. These are the '?' and '\*' characters. The '?' represents any character in the given position. The '\*' represents any or no character in the given position and in the rest of the positions of the filename or extension. In practice, the '\*' is used often, while the '?' is rarely used.

<sup>4</sup> Chapter 4 - Menu Program.

To explore wildcards we need some more files on the disk. Type in the following lines shown on the screenshot:

```

D1: COPY TEST.DAT ABCDE.DAT
TEST.DAT

D1: COPY TEST.DAT ABZDE.DAT
TEST.DAT

D1: COPY TEST.DAT ABCRAIG.DAT
TEST.DAT

D1: COPY TEST.DAT TEST.DOG
TEST.DAT

D1: COPY TEST.DAT TEST.DZT
TEST.DAT

D1: COPY TEST.DAT ABCDE.ICD
TEST.DAT

D1: █

```

Fig. 10: Copy Files

Every copy process is confirmed by showing the name of the source file to be copied. Earlier it was mentioned that some commands use default parameters, if none are provided. DIR is one such command.

The default 'fname.ext' (the one that is assumed if none is entered) is '\*.\*', meaning a file that has any or no characters for the name and any or no characters as the extension. Obviously this would include any files, so a complete directory is displayed.

What this means is that you can add a file name and extension to the DIR command to get a partial listing from the directory.

Type **DIR TEST.DAT**.

Since only one directory entry matches that name, it is the only one listed.

Now try **DIR \*.DAT**.

Several files are listed, but only those with an extension of 'DAT'.

Now try **DIR A\*.D\***.

This shows only those files whose name starts with 'A' and whose extension starts with 'D'.

Now try **DIR AB?DE.DAT**.

The '?' means any character, so 'ABCDE.DAT' and 'ABZDE.DAT' will be selected. Play around with DIR and different file masks (like 'ADC\*.D?T' and anything else you can think of) until you feel comfortable with the concept of wildcards.

See the results of the above examples on the next page.

```

D1:DIR *.DAT
Volume: TESTDISK
Directory: MAIN
ABCDE DAT 256 25-01-86 20:15
TEST DAT 256 25-01-86 20:15
ABZDE DAT 256 25-01-86 20:15
ABCRAIG DAT 256 25-01-86 20:15
698 FREE SECTORS

D1:DIR A*.D*
Volume: TESTDISK
Directory: MAIN
ABCDE DAT 256 25-01-86 20:15
ABZDE DAT 256 25-01-86 20:15
ABCRAIG DAT 256 25-01-86 20:15
698 FREE SECTORS

D1:█

```

Fig. 11: Use Of Wildcards For Directory Listing

Wildcards should be carefully used with the commands ERASE and RENAME, since they easily erase or misname multiple files.

### 2.13 Directories

What you have seen so far when listing the disk directory is the main directory (it even says so at the top). SDX allows you to add other directories to the disk.

Picture the disk as a filing cabinet. When you want to access a file, you (or SDX) have to check the whole drawer until the file requested is found. This is no problem if there are not many files in the drawer. Just think how time consuming it would be, though, to have to search through a stack of 100 files every time you wanted a file from the drawer. There are also times when it would be useful to group similar files together, such as keeping all of your paint program picture files together.

Subdirectories can be thought of as folders in the filing cabinet. If you took all of your picture files and put them in a folder labeled 'PICTURES', then you would only have to search the drawer until you found the 'PICTURES' folder, then search it for the desired picture file. Similarly, you could place all of your BASIC programs in a folder named 'BASIC', your text files in a folder named 'TEXT', and so on. Then, instead of holding a large stack of loose files, your filing cabinet would hold a well organized collection of folders. Searching through them for the proper one would be much easier than checking every file.

Subdirectories may also be placed within subdirectories, as deeply as you desire. For example, you could create a 'WILDLIFE' and a 'CARTOONS' directory within the 'PICTURES' subdirectory.

Type in **MKDIR TESTS**. This means **MaKe DIRectory TESTS**. Now list the directory. You will see an entry marked as directory by '<DIR>'.  
 You have just created a subdirectory, or 'folder', named 'TESTS'. It is, however, empty.

To verify this, type **DIR TESTS>**.



```

ABZDE   DAT      256 25-01-06 20:15
ABCRAIG DAT      256 25-01-06 20:15
TEST    DOG      256 25-01-06 20:15
TEST    DZT      256 25-01-06 20:15
ABCDE   ICD      256 25-01-06 20:15
TESTS   <DIR>    25-01-06 20:16
      696 FREE SECTORS

D1:DIR TESTS>

Volume:   TESTDISK
Directory: TESTS

      696 FREE SECTORS

D1:COPY *.DAT TESTS>*. *
ABCDE.DAT
TEST.DAT
ABZDE.DAT
ABCRAIG.DAT

D1:ERASE *.DAT

D1:█

```

Fig. 12: Use Of Wildcards When Copying

COPY is another command that allows wildcards. The next step is to copy all of the files ending in '.DAT' from the main directory to the subdirectory TESTS. And thereafter those files will be deleted from the main directory. To achieve this please type the respective commands as shown in the previous screenshot.

List the directory to assure yourself that the '.DAT' files are gone. List the directory of the TESTS subdirectory as you did before. You have moved the .DAT files from the main 'stack' to the TESTS 'folder'.

### 2.14 The Current Directory

The current directory is the one that is assumed when none is specified. In all of the examples so far the current directory has been the main directory. This can be changed with a simple command.

Type **CHDIR TESTS**. This means **CH**ange **DIR**ectory.

Now type **DIR**.

```

D1:CHDIR TESTS

D1:DIR

Volume:   TESTDISK
Directory: TESTS

ABCDE    DAT      256 25-01-06 20:15
TEST     DAT      256 25-01-06 20:15
ABZDE    DAT      256 25-01-06 20:15
ABCRAIG  DAT      256 25-01-06 20:15
      696 FREE SECTORS

D1:MKDIR ANOTHER

D1:CHDIR ANOTHER

D1:CHDIR
\TESTS\ANOTHER

D1:█

```

Fig. 13: Use Of Change Directory

You will get the directory of TESTS, not the main directory. Now whenever this disk is referenced without a directory mentioned, you will get the directory TESTS. Create another directory and change the current directory to it.

The last CHDIR with no arguments shows the path from the main directory to the current directory. This means that you are in the directory ANOTHER which is in the directory TESTS which is in the main directory.

If you want to return to the MAIN directory, type **CHDIR \** or **CHDIR >**.

And to just get back one level towards the root directory from the current directory (i. e. from ANOTHER to TESTS in the example above), type **CHDIR ..** (yes, two periods).

More information on subdirectories is provided in chapter 3. While subdirectories are invaluable for owners of large capacity drives, they are generally not needed with standard floppy disk drives unless a large number of very small files are on a disk.

### 2.15 Running Programs

To run binary files from SDX you just type in the name of the file. For example, to run a program named 'BALLSONG.OBJ' just type in **BALLSONG.OBJ**.

If no extension is given, '.COM' is assumed. To run a program without an extension, then, it is necessary to follow the file name with a period. For example, if the name of the file was 'DEMO', you would have to type **DEMO.(period)**. If you left off the period, SDX would try to run a program named DEMO.COM.

### 2.16 BASIC, CAR, and X

As demonstrated earlier, to enter internal BASIC in the XL/XE machines type **BASIC**. To enter an external cartridge put on top of the SDX cartridge type **CAR**. SDX handles both, internal BASIC and external cartridge separately. On XL/XE machines you may use another programming language in an external cartridge and switch between it and internal BASIC without any hassles.

It is never necessary with SDX to hold down the <OPTION> key to disable internal BASIC while booting or to remove an external language cartridge. However, for programs that would ordinarily require the removal of these cartridges, it is necessary to use the X command. For example, 'Turbo-BASIC XL' (TBS.COM) will not run with any cartridges installed. To run it from SDX call its filename this way: **X TBS** or **#TBS**.

This will probably be necessary for most of your large binary load files, since few have been written to avoid cartridge memory. It is recommended to execute programs always using X.COM since it is the proper way of getting rid of the troublesome 'Memory conflict' error. If using X.COM does not cure that, you probably need to reconfigure the system<sup>5</sup>.

Hard disks users can configure DOS in a way that it automatically disables the ROM module while executing a program<sup>6</sup>.

---

<sup>5</sup> Chapter 8 - Configuring Your System.

<sup>6</sup> Chapter 8 - COMEXE.SYS for more details.

### 2.17 Building Batch Files

A 'Batch File' is simply a file containing a list of commands, one on each line, that you wish the computer to perform automatically. Each line contains the command exactly as you would type it in. A batch file can have any legal file name, but the extension '.BAT' is assumed. Batch files are executed by typing a hyphen followed immediately (no space) by the filename. For example, '-TEST' would execute the batch file 'TEST.BAT', while '-DO\_IT.TXT' would execute the batch file 'DO\_IT.TXT'.

During the boot process, SDX will automatically execute a batch file in the main directory on D1: called 'AUTOEXEC.BAT', if exists. This allows you to have several commands executed every time you boot the computer. For example, suppose you need to setup the proper date and time since you do not have a real-time clock. Switch to the main directory of your boot drive. Now on the command line just type

```
ED AUTOEXEC.BAT
```

The editor bundled with SDX starts and opens up the file AUTOEXEC.BAT. Since you create a new file there will be an empty screen. The fail message in the head line occurs to notify the user that there is no file with this name yet and therefore could not be opened. But we are ready to create it now:



Fig. 14: Create New Batch File Using ED

Type in the needed commands with a <RETURN> at the end of each line as depicted on the screenshot below..

Next press <ESC> once and the screen should look like this:



Fig. 15: Save New Batch File In ED

- then <RETURN>. Your new AUTOEXEC.BAT will be saved now.

Type COLD on the command line and watch the execution.

Batch files are one of the most useful features of SDX. You can create them as well with any word processor or editor that will save a file as straight text (without formatting commands) or even the old-fashioned way utilizing the COPY command.

### 2.18 Practice

Now that you have a basic understanding of the operation of SDX and the command processor, the best thing to do is play around with the commands covered in this chapter and the rest of the commands found in chapter 4. With practice, you will soon have the most commonly used commands memorized and will feel very comfortable with the CP interface. In fact, the next time you boot with AtariDOS you may feel somewhat restricted by the menu!

### 2.19 DOS 2 Equivalents

The following is a list of the commands from the AtariDOS 2.0s menu and their equivalents in SDX:

<b>A - DISK DIRECTORY</b>	<b>DIR</b> and <b>DIRS</b>
<b>B - RUN CARTRIDGE</b>	<b>BASIC</b> for internal BASIC in XL/XE computers, <b>CAR</b> for an external cartridge
<b>C - COPY FILE</b>	<b>COPY</b>
<b>D - DELETE FILE</b>	<b>ERASE, DELETE, or DEL</b>
<b>E - RENAME FILE</b>	<b>RENAME, REN</b>
<b>F - LOCK FILE</b>	<b>ATR +P</b>
<b>G - UNLOCK FILE</b>	<b>ATR -P</b>
<b>H - WRITE DOS</b>	not needed - SDX boots from cartridge
<b>I - FORMAT DISK</b>	<b>FORMAT</b>
<b>J - DUPLICATE DISK</b>	<b>COPY, MENU</b>
<b>K - BINARY SAVE</b>	<b>SAVE</b>
<b>L - BINARY LOAD</b>	program name, <b>LOAD</b>
<b>M - RUN AT ADDRESS</b>	External command <b>RUN</b> from SDXTK
<b>N - CREATE MEM.SAV</b>	<b>SET CAR</b> and <b>SET BASIC</b>
<b>O - DUPLICATE FILE</b>	<b>MENU</b>

SDX comes with numerous commands that have no equivalent in AtariDOS 2 or any other DOS for that matter. It also supports any drives that are legal in the A8 system, ramdisks, time/date stamping, subdirectories, and more.

## 3 SpartaDOS X Overview

This chapter is an overview of file and path conventions, device identifiers, and general usage. It is assumed in this chapter that you already have a working knowledge of the command processor. Please refer to chapter 2 if you get confused.

You will find that many of the features described here are new to the A8. SDX is far more advanced than any other DOS for A8 computers.

A simple example: SDX allows drive identifiers like 'A:' and detailed pathnames like '>DOS>SUB2>MYPROG.BAS'. It supports all of the common hardware from today's as well as from the good old days of home computing.

### 3.1 Drives

Originally, most users of A8 computers utilized a floppy disk drive as the one and only mass storage medium. Very few users used to have hard drives with their Atari 800, XL or XE back in the heydays of 8-bit-computing. Today this is quite different, since many users engage emulated mass storage media, hard drive interfaces with hard drives, CF or SD card adapters and even emulators on other platforms. To simplify this guide the terms 'drive' and 'medium' incorporate all of this, if not noted otherwise. If you come across a problem or incompatibility, please send us a note at [dlt@atari8.info](mailto:dlt@atari8.info) or report it in the corresponding thread at <http://www.atariage.com/>.

### 3.2 Filenames

The basic form of the filename is identical to SD 3.2 - it consists of a name and an optional extension separated by a period. Legal characters are as follows

The letters 'A' to 'Z' - lowercase letters are converted to uppercase letters.

The digits '0' to '9' - filenames may start with a digit.

The underscore character ('\_')

The at character ('@')

All other characters are not allowed and may create problems, if read from media written to by other DOS or programs.

Throughout this manual, we use 'fname.ext' to represent a filename. The 'fname' portion may be up to 8 characters in length, and the 'ext' portion may be 0 to 3 characters in length and is optional.

### 3.3 Filename Extensions

It is important to develop a standard for naming files. The most common method is to reserve specific extensions for certain types of files. The following list contains some of the most commonly used extensions and their corresponding file types.

.ACT	An ACTION! source program
.ARC	A compressed archive of one or more files
.ASM	An ASCII machine language source file
.BAS	A BASIC SAVED program

.BAT	A (Sparta)DOS batch file
.BXL	A BASIC XL SAVEd program
.BXE	A BASIC XE SAVEd program
.COM	A (Sparta)DOS external command or binary program
.DAT	A data file
.DOS	A disk-based version SD
.EXE	A file or program being executable
.LST	A LISTed BASIC program
.M65	A MAC/65 SAVEd machine language source file
.OBJ	A binary object code file
.PRN	A listing to be printed
.SYS	A (Sparta)DOS system file or driver
.TUR	or '.TBS' is a TURBO-BASIC SAVEd program
.TXT	An ASCII or ATASCII text file, sometimes also ASC

In some cases the extensions will be assumed by the command processor or application. For example, '.COM' is assumed for command programs, '.EXE' for executables, '.BAT' for batch files, '.SYS' for drivers, and '.ARC' for archives.

### 3.4 Wild Card Characters

Two wildcard characters ('\*' and '?') can be used to take the place of characters in a filename in order to represent a range of filenames. The question mark ('?') is a 'don't care' character - it will match any character in its position. The asterisk (\*) in a filename or extension indicates that any or no character can occupy that position and all remaining positions in the filename or extension. Please refer to chapter 2 for more details and examples.

### 3.5 File Attributes

The SDFS already allowed attributes with previous disk based versions of SD. SDX adds two attributes to the standard SDFS directory entry - these are the hidden and archived bits - and a tool<sup>7</sup> to handle all of them.

While the protection and hidden bits work like known from other operating systems the archived bit works different, depending on the referenced OS. This bit is set whenever a file had been successfully archived by a back up program such as 'FlashBack!' or the SDX command COPY.

Another attribute with the SDFS is the subdirectory bit. It enables the system to distinguish between files and folders (or subdirectories) when identifying directory entries. An attempt to change this status bit would corrupt the subdirectory integrity. Data loss is the likely outcome of such an action.

SDX accesses attributes either in scan or in set/clear mode depending on the command used to work with.

For more information on directory entries please refer to the 'Technical Information'.<sup>8</sup>

---

<sup>7</sup> Chapter 4 - ATR.

<sup>8</sup> Chapter 7 - Technical Information.

### 3.6 Time/Date Stamp

Another feature SD brought to A8 is the time/date stamp. It documents the last write access to a file for future reference to enable the user to keep track of the age of different versions of files on several media. With subdirectories only the creation time/date is registered. It will not change when files within a subdirectory are created or updated. Once familiar with it on A8 it will become absolutely essential for the SDX user. Of course a proper clock setting is needed as prerequisite.<sup>9</sup>

### 3.7 Directories

The disk is broken up into directories, each of which may contain up to 1,423 entries. SD versions 1.x - 3.x have a limit of 126 entries, so they will not read SDX directories beyond the 126th entry! The root directory is named 'MAIN' and other directories (called subdirectories) may be created under 'MAIN'.<sup>10</sup> For directory names the same rules apply as for filenames; they may have an extender.

When displaying a directory, subdirectories will appear in the listing with a '<DIR>' in the file size field. Subdirectories may be nested with no limits other than disk space and practicality.

**Notes:** While SDX will support up to 1,423 entries in each directory, it is recommended that you try to avoid having more than ca. 200. The size of the directory must increase to allow additional files, and, once increased, will never decrease. A directory holding 1,423 entries would be 32 KiB in size. Large directories will slow down disk access considerably, especially when opening new files.

### 3.8 Pathnames

SDX uses a path to describe the route from one directory to another. The characters '>' or '\ ' are used as directory name separators. If used at the beginning of a path, they tell SDX to begin at the root directory (MAIN). Also, if one or more '<' characters (or '..\ ' strings) begin a pathname, SDX will back up one level toward the root directory for each occurrence. To be more precise, the pathname consists of the path and the filename. Here are some sample pathnames

```
>DOS>CHTD.COM
\DOS\CHTD.COM
TEMP>JUNK>TEST.DAT
<EXPRESS>EXPRESS
..\EXPRESS\EXPRESS
```

The first two are equivalent - from any directory they both access the file 'CHTD.COM' in the 'DOS' subdirectory of 'MAIN'. The third example accesses the file 'TEST.DAT' in the subdirectory 'JUNK' which is in the subdirectory 'TEMP' which is in the current directory. The fourth and fifth are equivalent - they both access the file 'EXPRESS' in the subdirectory 'EXPRESS' which is in the parent directory of the current directory.

**Note:** Since '<<' is used by SDX for input redirect, it is necessary to precede two or more '<' characters with a colon when they are intended as directory specifiers. The colon simply means the current drive and directory, but it keeps the '<<' off the front of

<sup>9</sup> Chapter 2 - Setting time and date.

<sup>10</sup> Chapter 4 - MKDIR.

the command line argument and prevents it from being interpreted as a redirect command. For example,

```
DIR <<PICTURES>
```

does not like in previous versions of SD give the directory of the pictures subdirectory found in the directory two toward the root. Use instead

```
DIR :<<PICTURES> or DIR ..\..\PICTURES\
```

The longest pathname SDX allows is 63 characters. This has no effect on the maximum number of levels of nesting, but does impose a practical limit of about 8 levels.

### 3.9 Command Length

The maximum accepted length of a line at the command line is 63 characters. No warning occurs when exceeding this limit. Additional characters will simply be ignored<sup>11</sup>. This 63 character limit includes the command name itself but not the prompt.

### 3.10 Filespec

Describes the specification of a file and its place within the SDFS. Device identifier and pathname make up the filespec. 'C:>TOOLKIT>UTILS>HDSC.ARC' as filespec describes the exact place of the file named 'HDSC.ARC' on drive 'C:' using the path '>TOOLKIT>UTILS>'.

### 3.11 Device Identifiers

In earlier versions of SD all device identifiers were based on the CIO device table. SDX is much more flexible. It has a second CIO type entry point (kernel) and provides device names for the same resources as the standard CIO. E. g. the standard I/O device (Editor) in the CIO is referred to as 'E:', but in the SDX 'kernel', it is referred to as 'CON:'.

There are a number of reasons to deviate from the standard Atari way:

- SDX was designed to feel exactly like an MS-DOS machine.
- The X cartridge ROM is a file oriented device (much like a disk), and needs a device specifier for it.
- The 'kernel' with a device system has to be entirely independent of the CIO IOCB tables anyway. There are simply too many technical problems in using the CIO when you need a high degree of flexibility.
- Referring to drives by either a letter or number, the letters would have conflicted with already existing devices.

And there are many more reasons - the before mentioned are just a few. So, here is what we have come up with:

On top the 'devices' as there are DSK, CLK, CAR, CON, COM, PRN, NUL, PCL. Three of them are file-oriented: DSK, CAR, PCL.

---

<sup>11</sup> Chapter 8 - DOSKEY.SYS to get a stop function.



One level below there are 'units'. In theory, every device can have up to 15 units, although most of the devices contain only one. Devices with multiple units currently are DSK, NUL, PCL.

The units of the DSK device are the drives.

One step down is the SIO level. The SIO has its own set of devices and units, like DSK1.

Also at the SIO level the calls are dispatched between serial devices (SIO proper) and parallel devices (PBI). For the all the above layers the SIO and PBI devices are not distinguishable, they behave the same way and share a common SIO API. Just here is the point of divergence.

One step below there are actual low level communication drivers, e. g. such as the one which resides in a hard disk interface ROM.

Through the command processor, the devices are as follows:

DSKx: 'DSK' is the official device identifier for your drives - since it is assumed, there is no need to type it.

CLK: is the registered clock device, realtime or software clock.

CAR: 'CAR:' is the X cartridge 'ROM disk' - you may load files from it or do directories of it, but of course you may not save or write to it.

CON: 'CON:' is the standard I/O device (in prior SD versions it was called 'E:')

COM: 'COM:' is the RS232 port (again, you may follow 'COM' with a port number). There is no default driver for this device.

PRN: 'PRN:' is the printer (you may follow the 'PRN' with a printer number 1-4 or A-D).

NUL: NUL: is a device, that can accept any amount of data written to (not saving it anywhere), and, while reading from it, it can behave in three different manners:

'NUL:' or 'NUL1:' returns error 136 (EOF).

'NUL2:' returns an infinite number of zeros.

'NUL3:' returns an infinite number of random bytes.

PCL: A PCLink device allowing the DOS to act as a client of a file server using the DOS2DOS protocol. There is no default driver for this device.

SDX can handle up to fifteen drives attached simultaneously to the A8. At SIO level the drives are numbered from 1 to 15 (\$01-\$0F). Drives 1-9 have, just as in SDX versions before 4.40, identifiers 1: to 9: or A: to I: in the Command Processor, and D1: to D9: or DA: to DI: in BASIC. The drives 10-15 have letter identifiers only: J: to O: in the Command Processor, and DJ: to DO: in BASIC.

A: ... O: The letters 'A' through 'O' represent the drives 1 through 15 when used without a device name (a three letter name) in front - the device 'DSK' is

always assumed if none is specified. Lower case is treated as if it were upper case - always!

- 1: ... 9: The numbers '1' through '9' represent the drives 1 through 9 as above - a '2:' is absolutely identical to a 'B:'
- Dx: A single 'D' (or 'd') preceding a letter or number is simply ignored. (Thus 'D2:' or 'Db:' means drive 2 as always.)
- D: No, this is not the current drive or drive 1 - it is drive 4!
- : Since there is no drive letter, this is the current drive.

Now you may ask, 'How do I access these devices through the CIO in BASIC?' Well, this is simple - precede the device or drive number by a 'D'. Here a few examples to emphasize the point.

```
OPEN #1,4,0,"D:README.DOC"
```

opens the file 'README.DOC' from the current drive, and

```
OPEN #1,6,0,"DCAR:*.*"
```

opens the X cartridge directory.

The command

```
LOAD "DB:TEST.BAS"
```

loads the program 'TEST.BAS' from drive 2, and

```
LIST "DPRN:"
```

lists your program to the printer (of course you could have used 'P:' instead of 'DPRN:').

The given examples show how to use the SDX 'kernel' through the CIO 'D:' device. Of course you still have all the other standard CIO devices at your disposal (e. g. 'E:', 'P:', 'C:', 'K:', 'S:', etc.). The point is that through the command processor you may only access the SDX 'kernel' devices, but through the CIO you may access both sets of devices.

### 3.12 Default Drive and Directory

The default drive and directory are the drive and directory the system uses when none are specified. Each drive has a default (or current) directory. To change the default drive, at the DOS prompt, simply type the new device identifier followed by a <RETURN>. For example

```
C:
```

sets the default drive to drive 3.

To change the current directory on a drive, use the command CHDIR (or the short forms CWD or CD). For example

```
CHDIR DOS
```

sets the current directory on the default drive to 'DOS', and the command

```
CHDIR B:BASIC
```

sets the current directory of drive 2 to 'BASIC' (assuming that the directories DOS and BASIC in these examples exist in the current directories, respectively).

### 3.13 Volume Names

All SDFS formatted media have a volume name. They are used for two reasons:

- To better organize your media by naming them.
- For SD to quickly tell the difference between e. g. the current disk and the next disk you put in the drive (there is no way to tell when the door opens on the drive).

If you display a directory of a medium in an AtariDOS 2 compatible format, you will find that they are always named 'DOS 2.0' so that you can quickly tell the difference between media formats. Also, because there is no unique volume name on AtariDOS formatted media, the buffer system does not remember anything about the last access to these media. Therefore, AtariDOS formatted media are much less efficient and slower. In addition, loading binary files, especially those with many segments, can take considerably longer from an AtariDOS medium than from a SDFS medium. It is recommended that files be copied to SD media or ramdisks before running.

### 3.14 Disk Format Compatibility

SDX 4.4x has no problem reading from and writing to media formatted and used with SD 2.x, 3.x, 4.1x and 4.2x. It also reads a medium formatted with SD 1.1. It is, however, not recommended to do any write operations (including file deletions) on SD 1.1 media, should they remain accessible to SD 1.1 afterwards. It is recommended to copy the files over to a SDX medium first, and edit them thereafter.

SDX 4.4x is able to build file systems on media formatted to 512 bytes per sector. This density is called DD 512. Files saved on such a medium are not accessible with any other version of SD, and, as far as we know, neither to any other DOS running on A8.

SD 2.x and 3.x will flawlessly read from or write to media formatted (except DD512) and/or written to by SDX. But there are restrictions: directory entries beyond the 126th will not be seen and may not be accessed. Deleting files before these in the directory will not allow them to be seen, since their physical position in the directory will not change.

SD 2.x and 3.x will ignore the new file attributes (Archived and Hidden).<sup>12</sup> They will not acknowledge these, nor will they change them.

---

<sup>12</sup> Chapter 4 - ATR.

### 3.15 Using External Cartridges with SDX

The legacy SDX 4.2x cartridge developed by ICD is a 'piggy back' cartridge, meaning that an external cartridge may be plugged into the top of the SDX cartridge. If you are running a different host for it, please check the respective manual on how to use external cartridges with it.

With a piggy back cartridge, if you are using a 800 computer, SDX should be plugged into the left cartridge slot in the computer. If you have a 130XE connected to a MIO, you may plug the SDX cartridge into either of the slots on the connector.

If you have a R-Time 8 cartridge, you can plug it in almost anywhere, since it is not recognized by the computer as being a cartridge. You can plug it into the left (or only) slot and plug SDX into it. You can plug it into the SDX cartridge. You can also plug it into the right slot on an 800 or into the extra slot on the 130XE/MIO adapter.

If your computer has only one cartridge slot, you may plug the SDX cartridge into the R-Time 8 or plug the R-Time 8 into the SDX cartridge.

Any external program cartridge must be plugged into the top of the SDX cartridge for the system to perform properly. Language cartridges such as ACTION!, MAC/65, etc. may be left in the top of the SDX cartridge until you wish to use some other cartridge. SDX allows you to enable and disable these cartridges by command. You can even turn both, SDX and the external cartridge, off to boot a game or another DOS without removing either cartridge.

Most game cartridges take control of the system during the boot process, preventing SDX from initializing. This will not keep the game from operating properly, but it will keep you from using SDX while the game cartridge is installed.

**Note:** Some SDX hosting platforms do not have an option to plug in a cartridge. If you like to experiment with it, see the emulator Altirra.

### 3.16 Drivers

A driver is a program that interacts with a device or another piece of software. Drivers are optional and carry information that the programs using the driver do lack. The SDX system comprises of a lot of drivers. Some of them are mandatory to run SDX as e. g. SPARTA.SYS and SIO.SYS, other drivers are optional, e. g. for individual hardware or software to be used like real time clocks or ramdisks.

Drivers for SDX are very specific and cannot be run on other operating systems than SDX. The way they are created has an advantage though as they can be loaded in two ways:

- During boot processing by a device call in a configuration file, e. g. DEVICE CON64, or
- manually from the command line interface,, e. g. CON64.SYS<RETURN>.

The latter one enables the user to temporarily load a driver that is not called during the boot process.

### 3.17 Memory Management

Since the 1980s there is a standard in addressing extended memory on A8 machines. It is called bank switching, meaning that additional memory beyond main memory is selected in banks of 16 KiB in size, which temporarily get 'switched in' into a 16 KiB bank of main memory.

As of version 4.47 SDX takes advantage of this in a way, that allows the memory handler to assign memory indices to all extended RAM banks. Now the bank switching of extended memory is completely hardware-abstracted. So there is no longer any need to know for SDX and its utilities what extension type (PORTB or Axlon) is dealt with.

The indices \$00 to \$03 are the same as in older versions:

- \$00 is main memory,
- \$01 is reserved and not to be used (like in older versions),
- \$02 is 'system extended area' (i. e. the area where SPARTA.SYS was loaded to),
- \$03 is reserved and not to be used (like in older versions).

All other indices from 4 onwards are assigned to 'program extended area', i. e. banks of extended RAM. To each bank found - e. g. 0-3 with a stock 130XE - a value of 4 will be added, creating the indices 4 to 7.

Other examples: 320 KiB - 16 banks, index 4 to 19  
576 KiB - 32 banks, index 4 to 35  
1 MiB - 64 banks, index 4 to 67  
4 MiB - 252 banks, index 4 to 255 (4,032 KiB in total)

SDX does not differ between Port B type or Axlon type, nor does it make any differences between Port B types of extended memory allowing separate access of CPU and ANTIC (Compy Shop) or extended memory which does not provide this feature (Rambo). SDX just looks up how much extended memory is available, sets the indices and configures it according to the user's settings in CONFIG.SYS.

SDX as of version 4.47 utilizes the new memory management to engage free banks of extended memory. It will be noted with the respective commands, tools, utilities, etc.

**Note:** If a XL/XE machine is equipped with a memory extension that uses bit 1 of PIA's port b, it will cause problems when the same bit is used to enable/disable the internal BASIC while accessing extended RAM at the same time. This is a hardware issue which cannot be solved using SDX.



## 4 The Command Processor - Commands

The description of the command processor is broken down into two chapters. The first - this one - is a listing of SDX commands in alphabetical order. The description of each includes the purpose, syntax, and type of command. The second ('The Command Processor - Advanced Features') discusses batch files and I/O redirection, and contains detailed information on some of the more complex features.

Command names and parameters represent their function or purpose so they should be easy to remember. The '**Purpose**' of each command is briefly defined so you quickly get an idea of what it is used for. Next the '**Syntax**', that shows the proper use of the command with its options, if applicable. These conventions are used in the '**Syntax**' section:

- [...]** Parameters in brackets are optional.
- a|b|...|z** One or more of these options may be selected. Refer to the specific command's remarks for details.
- symbol** Some commands offer to input a symbol in the parameter section as a substitute to hex or decimal values.
- device** Available/installed kernel device (DSK:, CLK:, CAR:, CON:, PRN:, NUL:).
- d:** Drive number or letter (A:...O:, 1:...9:, D1:...D9:, etc.).
- path** The path from the current or root directory to the desired one, such as SDX>TOOLKIT> or \DOS\
- fname** The one to 8 character filename. With many commands, wildcards (\* and ?) are allowed. Refer to the remarks for specifics for each command.
- .ext** A 0 to 3 character file extension. Wildcards are often allowed.
- +, -, /** These characters should be used as shown.

If there is an 'Alias' for the command, it is shown next. We tend to have an alias when there is a shorter form, which seems logical. This way SDX remains compatible with older SD versions. Additionally, we try and maintain command similarities for people who are familiar with MS-DOS.

The 'Type' will either be 'internal' or 'external'. Internal commands are internal to the command processor itself - they require no other program to perform the command. External commands are found in the 'CAR:' device or may reference one of the files. Most of the SDX cartridge content is devoted to these external commands. Additionally, external commands may be performed from other sources, e. g. a disk or hard drive partition.

'Related' commands are noted next. These may be in the same class or family of commands, or may include other ways of accomplishing the same function.

'Availability' tells you from what version of SDX 4.4x on this command has been made available. Since there are several versions of SDX - old and new ones - it helps to distinguish them. Commands without this remark were already distributed in before with the ICD version of SDX.

'Remarks' includes all the details and special rules of command usage. The remarks may also show usage examples. A good way to learn SDX is to read through each command thoroughly and then try typing in examples of the command including its options. This will help you to understand SDX and enables you to become a SDX power user.

Please note that any numeric values can be either decimal or hex, but hex values must be preceded with a \$.

If you have used a prior version of SD, you will find the command processor similar in feel and will recognize most of the commands. Also, you will notice that the command processor has been greatly enhanced with more sophisticated batch files, command line I/O redirection, user definable prompts, command search paths, and more.

To ease the use of SDX there are help files in form of man pages available to be handled by the 'MAN' command. Many platforms for SDX provide sufficient memory to store these man pages as a kind of online help system in the CAR: device. They are also part of the SDXTK.

The SDXTK<sup>13</sup> enhances the SDX system by more external commands, drivers, tools and utilities. It is recommended to add it to your system environment.

More information and support can be found at:

<http://sdx.atari8.info/>

Now, on to the commands . . .



## 4.1 APPEND - add destinations to system path

---

### Purpose

Append the given drive and/or path at the end of the \$PATH variable.

### Syntax

APPEND d: | pathname | device

### Type

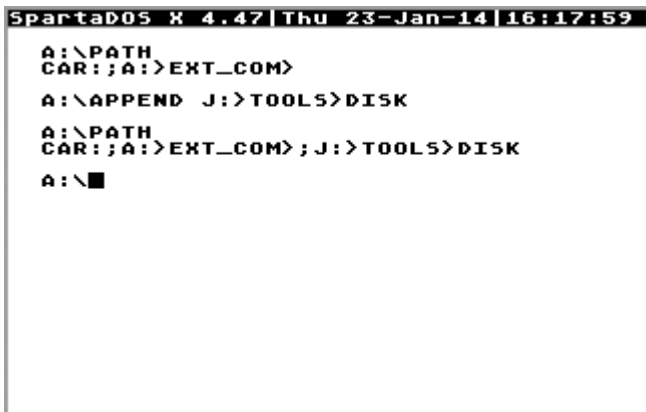
External - on CAR: device.

### Availability

As of SDX 4.40.

### Remarks

Sometimes it is very handy to have the path variable changed temporarily. Especially while programming and/or administering your system equipped with mass storage devices like hard drives, flash cartridges, SIO2XX devices etc. The APPEND command facilitates the task of temporarily adding a device, drive or directory to the \$PATH without manually rewriting all the paths that \$PATH contains.



```
SpartaDOS X 4.47 | Thu 23-Jan-14 | 16:17:59
A:\PATH
CAR: ;A:>EXT_COM>
A:\APPEND J:>TOOLS>DISK
A:\PATH
CAR: ;A:>EXT_COM> ; J:>TOOLS>DISK
A:\■
```

Fig. 16: Append A Temporary Path

## 4.2 ARC - archive files

---

### Purpose

Create and maintain file archives.

### Syntax

ARC command[option] [d: | device][path]arcfname[.ext] [filelist]

### Type

External - on CAR: device.

### Remarks

ARC, based on and compatible with ARC by System Enhancement Associates, will take a group of files, quickly combine and automatically compress them into a single archive file, taking up far less disk space. It will also add or extract files to or from this archive, show a directory of the archived files, display the contents of an archived file, show the compression method used, encrypt/decrypt files, and more. 'ARC' with no parameters will display a short description. ARC is now able to fetch files to be archived from any file-oriented device, not only from regular disks (DSK:) as before.

The 'arcfname' is the file name of the archive.

The 'filelist' is the list of files to be added, deleted, updated, extracted, etc., to or from the archive. Leave a space between each filename in the file list. Wildcards are perfectly legal. If no file list is entered, '\*.\*' is assumed. You may add a device and/or path to each filename, if needed.

'Command' is one of the following:

A Add file(s) to the archive. Add all files from the file list to the archive. E. g.,

```
ARC A DOCS A:\PRG\*.DOC C:\TOOLS\*.TXT
```

will generate the archive file 'DOCS.ARC', grab all DOC files from the subdirectory PRG on drive A: and all TXT files from subdirectory TOOLS on drive C:, compress them automatically and add them in alphabetical order to the archive file.

M Move file(s) to the archive. Move deletes the source file once it has been added to the archive.

U Update file(s) in the archive. It will look at the date of the files in the archive, replace older files with those of a newer date, and add all files (from file list) which do not currently exist in the archive.

F Freshen file(s) in archive. It is the same as update but without the 'add' feature. Freshen will replace the files in the archive with any newer files of the same name.

D Delete file(s) from the archive. It will remove the files listed in the file list from the archive.

X,E Extract file(s) from the archive. Both allow you to extract files. The method(s) of file compression used when creating the archive is reversed and the files specified in

the file list are restored to their original state. Add destination device/path if needed. E. g.,

```
ARC XH DOCS B:\READ\TXT\*.TXT F:\PRINT\*.DOC
```

will extract from the archive file 'DOCS.ARC' all TXT files and write them back to sub-directory READ on drive B., and extract all DOC files and put them into subdirectory PRINT on drive F.:

If the filename of a file to be extracted from an archive already exists on your destination drive, you will be prompted with

```
File exists - Overwrite (Y/N/A) ?
```

'Y' will do, 'N' won't do (nothing will be extracted) and 'A' stands for 'Overwrite all' and do not ask if to overwrite every single file. This prompt will not appear when extracting an archive to NUL.:

- P Print file(s) to the screen. This allows you to examine the contents of files within an archive without extracting them. Of course this can be diverted to other devices with redirection. E. g.,

```
ARC P MYARC READ.ME >>PRN:
```

will divert the contents of 'READ.ME' from the archive 'MYARC' to your printer.

- L List file(s) in archive. This shows the filename, original file length, and date/time stamp of each file in the archive as well as the number of files and total size of files if extracted. Use a filemask as shown with the V command to see all files.
- V Verbose list of file(s) in archive. It shows the filename, original file length, number of files, and total size, just as the L command does. Instead of date and time created, however, the V command shows stowage method, stowage factor (percent of space saved), the file size now, and the total size now. If the screen is wide enough with 64 columns or more<sup>24</sup>, the complete listing is displayed. E. g.,

```
ARC V FOOBAR
```

lists all the contents of the file FOOBAR.ARC, whereas only COM files whose names start with A are listed doing this:

```
ARC V FOOBAR A*.COM
```

**Note:** 'Total's in ARC V are 24-bit, thus they overflow, when e. g. the original length exceeds 16 MiB.

Valid '**Options**' are:

- S Suppress compression. This will archive files without compressing them. Most people will not use this option but it is faster than using compression.

- B Retain a backup copy of the archive. This is a safety option for the A, M, U, F, and D commands. The B option will result in a backup of the old archive with the extension of '.BAK' as well as the new archive.
- W Suppress warning messages. Use this command sparingly if at all. This will prevent those unsightly errors from being displayed but will also prevent mistakes from being discovered and avoided.
- N Suppress notes and comments. This will suppress the display of the standard ARC screen output which shows the current file being compressed or extracted, the compression method used, etc.
- H High speed. With the screen off on the Atari, processing speed is increased 20% to 30%. If you wish to go faster but don't need to see the screen, use this option. The screen display will return when finished, an error occurs or the overwrite prompt appears.
- G Encrypt/decrypt an archive entry. This prevents others from reading your files. G must be the last option and must be followed by a password. Without the password the archive entry can be extracted encrypted. E. g.,

```
ARC AHGICD STUFF DOIT.DOC DOIT.COM READ.ME
```

will add these three files in the file list into the archive called 'STUFF.ARC' under the password of 'ICD' with the screen off.

Archive entries are always saved in alphabetical order. This sorting function puts a practical limit of about 80 files per archive on 64 KiB machines (USE OSRAM<sup>15</sup>) and 180 files per archive on computers which use the extended memory mode (USE BANKED). ARC will not run on 48 KiB machines unless they have an AXLON compatible memory upgrade installed. Archive entries do not save pathnames, so avoid duplicate file names (one will replace the other).

ARC is very useful for saving time while uploading/ downloading files and saving space for archival storage. It uses four stowage methods and automatically determines the best method(s) suited to each file. The SDX version of ARC is fully compatible with ALF-crunched files, but it is highly recommended that you unARC an ALFed file and then ARC it before adding or updating. This will assure the most compact compression and arrange all files alphabetically within the archive.

These stowage methods are used:

Stored - no compression used. This is mainly used with very short files.

Packed - Strings of repeated values are collapsed. All files are packed before other compression methods are attempted.

Squeezed - Huffman encoding compression. This is usually only effective with larger machine language files. Huffman encoding uses a weighted binary tree method assigning the lowest bit representations to the most commonly used characters.

---

<sup>15</sup> Chapter 8 - CONFIG.SYS File & USE.

Crunched - Dynamic Lempel-Ziv compression with adaptive reset. This is created on the fly and is stored as a series of bit codes which represent character strings. Crunched is one of the more effective methods used. ALF-crunch exclusively uses a variant of this method.

**Notes:** It is a good idea to practice a bit to become acquainted with the power and capabilities of ARC. Even today ARC still has its advantages.

### 4.3 ATR - file attributes

---

**Purpose**

Set/clear file attributes.

**Syntax**

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

**Alias**

ATTRIB

**Type**

Internal

**Related**

DIR

**Remarks**

The scope of the ATR command is all files and folders (subdirectories) matching the given name. This command works not in scan mode, but in operation mode. Therefore the attributes for the directory entry matching the given name specification will be set(+) or cleared(-).

The directory entry attributes are as follows:

- A Archived - Files and Folders are marked as archived when this bit is set. Additionally, the archived bit is set when the file is backed up by a program such as 'FlashBack!' or the COPY command. This attribute is cleared whenever a file is updated or created. This attribute is not related to the ARC command.
- H Hidden - Files and Folders are hidden when this bit is set. A hidden file may be loaded as a command. Some commands such as TYPE and COPY will not see hidden files and/or subdirectories (unless attributes are specified with those commands).
- P Protected - Files and Folders are protected when this bit is set. They may not be deleted, overwritten or updated.

For example, to set the archived status and clear the protection bit of all '.COM' files, type the command

```
ATR +A -P *.COM
```

For further information about which status bits in the directory entry are affected by these attributes, please refer to the 'Technical Information' chapter.

**Note:** The attribute S (subdirectory) is unchangeable - thus not legal in the ATR command! This bit is set to indicate a subdirectory. If cleared, it would be seen as a file which could cause significant damage to the data on the medium.

## 4.4 BASIC - enter internal BASIC

---

### Purpose

Enter the internal BASIC in a XL/XE machine.

### Syntax

BASIC [/I|N] [d:][path][fname] [parameters]

### Type

External - uses CAR.COM on CAR: device.

### Related

CAR, SET

### Remarks

Without a filename the control is given to the internal BASIC of your XL/XE computer (except 1200XL). If a filename is specified, the internal BASIC will be enabled and the binary file you specified will be loaded and run. The optional 'parameters' are whatever the program 'fname' needs. The '/N' option returns to BASIC after running 'fname', instead of the command processor which is the default.<sup>16</sup>

To automatically load and run a BASIC program from the command processor, read about I/O redirection.<sup>17</sup>

SDX has a MEM.SAV facility similar to AtariDOS 2, but much more powerful. The environment variable \$BASIC should be set to the file you wish to use as the memory-save file for BASIC. If no such environment variable exists, this feature is disabled.

The \$BASIC variable has no default value, unless RAMDISK.SYS<sup>18</sup> is installed. If \$BASIC has not been set by the user<sup>19</sup>, the ramdisk driver sets the variable during installation so that it points to a BASIC.SAV file residing in the ramdisk.

You may change this with the SET command in CONFIG.SYS, AUTOEXEC.BAT or directly via command line. For example:

```
SET BASIC=D8:BASIC.SAV
```

sets the variable to 'D8:BASIC.SAV'. To see the current value of \$BASIC (and all the other environment variables) just type:

```
SET
```

To clear the variable, which disables the BASIC memory-save feature, just type

```
SET BASIC
```

---

<sup>16</sup> Appendix E - 'Using AUTORUN.SYS Files' for details.

<sup>17</sup> Chapter 5 -Advanced Features.

<sup>18</sup> Chapter 8 - Block I/O drivers.

<sup>19</sup> Chapter 4 - SET for more details.

Entering BASIC while no BASIC.SAV file exists, will result in a cold entry, meaning there will be no program in user memory. To force this, even when having a proper SAV file, you may use the '/I' option. The SAV file will be skipped then. Leaving to DOS next, will cause an overwrite of the old SAV file.

While the memory save feature is enabled and a problem loading or saving the memory file occurs (BASIC.SAV), an error message will be displayed. If this happens while loading the memory file, you will be prompted with the old MEMLO (when the file was saved). If an error exists while saving the memory file, you will be notified of that. In either case, you will have the option to abort and correct the problem or to proceed, deleting the memory file. Press <ESC> if you wish to abort or <RETURN> to proceed.

More details about the two situations that can occur:

- Upon entering BASIC, the current MEMLO does not match the MEMLO in the memory-save file. This can occur after installing extra drivers since last time you entered BASIC (such as the keyboard buffer, ramdisk, etc), or LOADING commands such as X or COMMAND<sup>20</sup>. At this point you may press ESC and restore the system to the way it was when you last entered BASIC (By COLD starting and/or LOADING programs), or press <RETURN> and enter BASIC cold. This will also happen if the memory-save file has somehow been corrupted.
- Upon exiting BASIC (using the DOS command) the system will attempt to create a .SAV file on disk. The disk is full, or is not online and the memory-save file can't be saved. You have the option to go to DOS (→ press <RETURN>) and lose the current BASIC program in memory, or to go back to BASIC (→ <ESC>) and SAVE whatever you were working on or clear up the disk problem.

In addition to saving the contents of user memory, the memory-save feature saves page 0 (from \$80-\$FF), and page 4-6. This means that you may alternate between BASIC and CARtridge without ever losing what you were working on. Whenever you enter BASIC the memory-save file is loaded, thus you can edit a program in BASIC, go to DOS, reboot the computer, and enter BASIC with exactly what you were working on before rebooting the system (as long the memory-save file is present and valid).

Performing a cold start (a jump to \$E477) while in BASIC will cause the SDX cartridge and the external cartridge plugged into the SDX cartridge, if any, to be disabled. This will have the same effect as typing COLD /N from the command processor.

**Notes:** This command is recognized by the command processor as an internal command that chains to the external program 'CAR.COM', so both the CAR and BASIC commands share the same external program. CAR.COM is memory resident while you are in the BASIC environment, so MEMLO will be slightly higher during this period.

If a XL/XE machine is equipped with a memory extension that uses bit 1 of PIA's port b, it will cause problems when the same bit is used to enable/disable the internal BASIC while accessing extended RAM at the same time.



## 4.5 BLOAD - binary load

---

### Purpose

Load a file to the specified memory area starting at the given address.

### Syntax

BLOAD [d:][path]fname[.ext] [[ $\$$ ]index:][ $\$$ ] address<sup>21</sup>

### Type

External - on CAR: device.

### Related

LOAD

### Availability

As of SDX 4.40.

### Remarks

The file filename.ext is loaded as a raw data block into the specified area, and then control is handed back to the Command Processor. The use of extended RAM to directly load binaries there is perfectly legal.

There are no checks done, whether the file fits in memory, or if vital OS areas are safe - it is assumed, that the user calls the command on purpose and is sure what he is doing.

Addresses are assumed to be decimal unless preceded by a '\$', which indicates hex.

---

<sup>21</sup> See Chapter 6.4 for new functionality in calculating an address.

## 4.6 BOOT - start a program at system start up

---

### Purpose

Tell a SDFS formatted disk/medium to boot a particular program at start up.

### Syntax

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

### Type

Internal

### Related

COLD, FORMAT

### Remarks

The DOS loader in the first three sectors of each SDFS 2.x formatted disk with sector size 128/256 bytes can load and run files in the same manner as a command file. Normally a DOS is loaded, but anything could be loaded as long as it avoids the loader memory (\$2E00-\$3180).

This command is the most misunderstood command in SD, so here are a few pertinent facts you should know:

- The BOOT command simply writes the starting sector number of the sector map of the file to boot in a specific location on sector 1 of the disk.<sup>22</sup>
- If the file which is set to boot is either ERASEd or COPYed over, the boot flag is cleared - the message 'Error: No DOS' will appear when attempting to boot that disk until a new file is set to boot!
- The file set to boot may reside anywhere on the medium - even in a subdirectory.

**Notes:** This command is very handy for installing a disk based version of a SDFS compatible DOS on larger media. Format it using SDX, copy the respective DOS onto it and make it bootable using the BOOT Command.

SDX does not write any DOS to the medium, it boots completely from cartridge.

Media formatted in DD 512 cannot be booted by the stock A8 OS. Therefore settings to boot a program from such formatted media will not work without a properly modified OS - a BOOT ERROR will appear.

BOOT does not work with media formatted using SD 1.1, since they do not contain a 3 sector boot routine! SDD 1.1 has a simplistic booting scheme which just loads a number of consecutive sectors.

Boot INIDOS.SYS<sup>23</sup> to re-enable a deactivated SDX cartridge.

---

<sup>22</sup> Chapter 7 - Technical Information.

<sup>23</sup> Chapter 9 - SDXTK.

## 4.7 CAR - enter a cartridge

---

### Purpose

Enter the cartridge plugged into the top of the SDX cartridge.

### Syntax

CAR [/F|N|I|L S|A|N|I|mode] [d:][path][fname] [parameters]

### Type

External - on CAR: device.

### Related

BASIC, COLD, SET

### Remarks

CAR invoked without any switches will hand over control to the cartridge plugged into the SDX cartridge. If a filename is specified, then that binary file is loaded and handled according to the given parameters. This is useful e. g. for compiled ACTION! programs that need to call routines within the cartridge. The optional 'parameters' are whatever the program 'fname' needs.

The /F switch (as Force) was implemented to support the Weronika cartridge. It causes the specified program to be executed even if there is no ROM at \$A000-\$BFFF (the Weronika cartridge can switch in own memory, but that is RAM, so the normal cartridge detection cannot detect it). Also, when /F is given, /N is ignored.

The '/N' option returns to the cartridge after running fname, instead of to the command processor, which is the default.

SDX features a MEM.SAV facility similar to AtariDOS 2, but much more powerful. The environment variable \$CAR should be set to the file to be used as the memory-save file for the cartridge. If no such environment variable exists, this feature is disabled.

The \$CAR variable has no default value, unless RAMDISK.SYS<sup>24</sup> is installed. If \$CAR has not been set by the user<sup>25</sup>, the ramdisk driver sets the variable during installation so that it points to a CAR.SAV file residing in the ramdisk.

You of course may change this with the SET command in CONFIG.SYS, AUTOEXEC.BAT or directly via command line. For example:

```
SET CAR=D8:CAR.SAV
```

sets the variable to 'D8:CAR.SAV'. And to see the current value of \$CAR and all the other environment variables just type:

```
SET
```

---

<sup>24</sup> Chapter 8 - Block I/O drivers.

<sup>25</sup> Chapter 4 - SET for more details.

To clear the variable (i. e. disable the CAR memory-save feature) type

SET CAR

Entering a CAR while no CAR.SAV file exists causes it to be cold initialized, hence there will be no program in user memory. To force this even having a proper SAV file, use the '/I' option forcing the SAV file to be skipped. Leaving to DOS next causes an overwrite of the old SAV file.

While the memory save feature is enabled and a problem loading or saving the memory file occurs (CAR.SAV), an error message will be displayed. If this happens while loading the memory file, you will be prompted with the old MEMLO (when the file was saved). If an error exists while saving the memory file, you will be notified of that. In either case, you will have the option to abort and correct the problem or to proceed, deleting the memory file. Press <ESC> if you wish to abort or <RETURN> to proceed.

More details about the two situations that can occur:

- Upon entering the cartridge, the current MEMLO does not match the MEMLO in the memory-save file. This can occur after installing extra drivers since last time you entered the cartridge (such as the keyboard buffer, ramdisk, etc), or LOADING commands such as X or COMMAND.<sup>26</sup> At this point you may press ESC and restore the system to the way it was when you last entered CAR (by COLD starting and/or LOADING programs), or press <RETURN> and enter the cartridge cold. This will also happen if the memory-save file has somehow been corrupted.
- Upon exiting CAR (using the DOS command) the system will attempt to create a .SAV file on disk. The disk is full, or is not online and the memory-save file can't be saved. You have the option to go to DOS (press → <RETURN>) and lose the current program in memory, or to go back to CAR (→ <ESC>) and SAVE whatever you were working on or clear up the disk problem.

In addition to saving the contents of user memory, the contents of page 0 (from \$80-\$FF), and page 4-6 is saved. Therefore you may alternate between BASIC and CARtridge without ever losing what you were working on. Whenever you enter the cartridge the memory-save file is loaded, thus you can edit a program in the cartridge, go to DOS, reboot the computer, and enter the cartridge with exactly what you were working on before rebooting the system (as long the memory-save file is present and valid).

Executing a cold start while in the cartridge will disable SDX, while leaving the external cartridge enabled. This is the same as typing COLD /C from the command processor.

The /L option allows advanced users to select the loading mode. It can be either the directly given number ("mode") or one of the following symbols:

- S - SDX loading mode (mode=0)
- A - AtariDOS loading mode (64)
- N - do not execute (128)
- I - do not execute, but initialize (192).

For details on the loading mode see the LOAD command.

If not specified, the loading mode is assumed to be 64 (AtariDOS). This will properly execute ACTION! compiled binaries, which set INITs but do not properly set the RUN vector.

**Notes:** CAR is recognized by the command processor as an internal command that chains to the external program 'CAR.COM', so both CAR and BASIC commands share the same external program. CAR.COM remains memory resident while in the cartridge environment, so MEMLO will be slightly higher during this time. It will return to the lower value when the cartridge is exited.

After a cold start the program from the SAV file created when using BASIC XE with its extensions will not be recovered. Make sure you saved it before executing a cold start.

## 4.8 CHDIR - change directory

---

### Purpose

Change the current (working) directory on the specified drive.

### Syntax

CHDIR [d:][path]

### Alias

CD & CWD

### Type

Internal

### Related

MKDIR, RMDIR, PATH, CDD (SDXTK)

### Remarks

Directories (also called subdirectories or folders) are used to organize your files. They also make searching large storage areas on hard drives much faster. In a file cabinet it is much quicker to go to a file folder and search through a few documents than searching through a pile of all documents. Computers work the same way. It is much quicker for DOS to go to a subdirectory and search through a few files than it is to search through one long file list. CHDIR allows you to move among your directories.

The current directory is where SD looks to find files whose names were entered without specifying a directory. If you do not specify a drive, the default drive is assumed.

CHDIR invoked with no parameters displays the current directory path of the current drive. Equals the '?DIR' command from SD 3.2.

The default directory of a drive is reset to the MAIN directory if the disk has been changed.

**Note:** This command has no effect on MyDOS disks even though subdirectories are supported. This is due to the fact that there is no foolproof way to detect a disk change on DOS 2 style disks. SD disks have a volume name, a random number, and a write count for disk change detection.<sup>27</sup>

## 4.9 CHKDSK - check disk

---

### Purpose

Show volume, free/total disk space, and sector size of the selected drive.

### Syntax

CHKDSK [d:] [/X]

### Type

External - on CAR: device.

### Related

FORMAT, MEM, VER

### Remarks

Typing 'CHKDSK' at the DOS prompt calls the program 'CHKDSK.COM' residing on 'CAR:' device. It is used to quickly see how much space is available on a drive and the sector size (this information is not available by doing a DIRectory). Note that the volume name of all AtariDOS 2 style disks will appear as 'DOS 2.0'.

The '/X' option causes an extended disk information to be displayed. This way you can quickly review the details of how your medium is set up.

**Notes:** The disk write-lock status known from earlier SD versions is no longer supported. It did not protect from formatting the disk.

The write-lock feature of the MIO<sup>28</sup> still works and is totally independent - it is a far more secure write-lock.

When using CHKDSK with the /X option, most of the displayed data are self explaining.

The two numbers following the volume name are used for disk change detection in cases where volume names are the same on both disks. The first is a sequence number which is incremented each time a file on the disk is opened for write. The second is a random number generated when the disk was formatted.

Date and time, when the disk was formatted, will be displayed at the end of the list, if this information is available.

## 4.10 CHTD - change time/date stamp

### Purpose

Change the time/date stamp on all files matching the given filespec to the current time and date.

### Syntax

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

### Type

External - on CAR: device.

### Related

DATE, TD, TIME

### Remarks

By default, this command will only change the time/date stamp on non-hidden and non-protected files – the attributes may be overridden.<sup>29</sup> A filespec must be entered since '\*' is not assumed.

```
SpartaDOS X 4.45 | Tue 8-Nov-11 | 15:46:43
D1:DIR
Volume: TESTDISK
Directory: TESTS
ABCDE DAT 256 25-01-06 20:15
TEST DAT 256 25-01-06 20:15
ABZDE DAT 256 25-01-06 20:15
ABCRAIG DAT 256 25-01-06 20:15
ANOTHER <DIR> 25-01-06 20:19
DATA ARC 1142 15-02-11 15:45
688 FREE SECTORS

D1:CHTD *.*
Changing TD in ABCDE.DAT
Changing TD in TEST.DAT
Changing TD in ABZDE.DAT
Changing TD in ABCRAIG.DAT
Changing TD in DATA.ARC

D1:█
```

Fig. 17: Change Time And Date Of A File

The switch '/Q' causes the command to suppress the message confirming the change of the timestamp.

<sup>29</sup> Chapter 4 - ATR for more information on attributes [A|H|P|S].



## 4.11 CHVOL - change volume name

---

### Purpose

Change the volume name on the specified drive.

### Syntax

CHVOL [d:]volname

### Type

External - on CAR: device.

### Related

CHKDSK, FORMAT, DIR

### Remarks

This command will not change the volume name on AtariDOS 2 disks since they physically have no volume name. Up to eight characters are allowed on SD formatted media. The volume name may contain any ATASCII characters including spaces and inverse characters. Leading spaces are not allowed.

```
SpartaDOS X 4.45 | Tue 8-Nov-11 | 15:47:18
D1:DIR
Volume: TESTDISK
Directory: TESTS
ABCDE DAT 256 15-02-11 15:46
TEST DAT 256 15-02-11 15:46
ABZDE DAT 256 15-02-11 15:46
ABCRAIG DAT 256 15-02-11 15:46
ANOTHER <DIR> 25-01-06 20:19
DATA ARC 1142 15-02-11 15:46
688 FREE SECTORS
D1:CHVOL TESTING
Volume Name changed to TESTING
D1:█
```

Fig. 18: Change Volume Name

## 4.12 CLR - clear variables

---

**Purpose**

Delete unused system variables.

**Syntax**

CLR

**Type**

Internal

**Availability**

As of SDX 4.42.

**Remarks**

Deletes the system variables, which were created by the system and are no longer used. This command is only necessary when the execution of a batch file was aborted with <RESET>. In such circumstances internal variables created by the batch file can remain in the environment area. The CLR command allows to delete them 'by hand'.

## 4.13 CLS - clear screen

---

**Purpose**

Clear the screen.

**Syntax**

CLS [/F]

**Type**

Internal

**Remarks**

Useful especially for batch files, CLS will simply clear the screen.

Using the /F option (as in 'force') will re-initialize the screen instead of clearing it. The code will perform 'GRAPHICS 0' rather than '? CHR\$(125)'. The SCRDEF values set in the AUTOEXEC.BAT will be re-enabled thereafter.

#### 4.14 COLD - cold start

---

**Purpose**

Reboot the system (by doing a jump through \$E477).

**Syntax**

COLD [/C|N]

**Type**

Internal

**Related**

BOOT, CAR

**Remarks**

This command is an alternative to switching the computer's power off and back on. The major advantage of using COLD is that the extended banks of RAM will retain their memory, thus the data in your ramdisks will still be there.<sup>30</sup> This is equivalent to the SD 3.2 command:

RUN E477

The command has two options:

- C - Reboot the computer with SDX disabled and the cartridge plugged into SDX enabled.
- N - Reboot the computer as if there were no cartridges in your computer.

Hold down <OPTION> while pressing <RETURN> to reboot without internal BASIC.

Generally, once SDX has been disabled, it will be necessary to turn the computer off and back on to re-enable SDX.

Practically, this hardware torture may be omitted by booting INIDOS.SYS<sup>31</sup>.

In the Maxflash versions of SDX 4.4x the 'COLD /C' command is an equivalent to 'COLD' alone (without the parameter).

---

<sup>30</sup> Chapter 8 - RAMDISK.SYS driver description.

<sup>31</sup> Chapter 9 - SDXTK.

## 4.15 COMMAND - The Command Processor

---

### Purpose

Enter commands and run other programs.

### Type

External - on CAR: device.

### Related

All Commands

### Remarks

It is not entered as a command itself but is automatically invoked when you enter DOS. Many commands are of type 'internal' - this means that the command processor knows how to perform the command without loading any other programs.

'External' commands must load from disk or CARtridge into memory and then perform their function. When you execute these commands, they must reside on the current drive and directory, otherwise you must specify what drive or device they reside on (by preceding the command with a device or drive identifier). The PATH command can add additional drives/paths to search for the file. For example the default PATH, which is 'PATH CAR:', allows commands such as CHTD or DUMP to run without having to specify the 'CAR:'. Of course you may add additional directory paths.<sup>32</sup>

You will notice that the command processor itself is 'external'. This is to give you more memory (3-4 KiB) to run your application programs. In fact, whenever you run an 'external' command or program, the command processor is unLOADed from memory and replaced by the new program. When that program is finished running, the command processor is reLOADed and awaits your command. The exception to this rule is if you enter the command

```
LOAD COMMAND.COM
```

This actually holds and links the command processor in memory, thus the unLOAD/reLOAD cycle is circumvented.

As of SDX V. 4.48 the user may switch to another default device than a disk. E. g. CAR: is now a legal default device. When the default device is not a disk, its identifier will be automatically displayed before the unit number (\$N) or unit letter (\$L).

See PROMPT<sup>33</sup> command for display options.

---

<sup>32</sup> Chapter 4 - PATH.

<sup>33</sup> Chapter 4 – PROMPT.

## 4.16 COMP - compare files

### Purpose

Compare two files.

### Syntax

COMP [d:][path]fname1.ext [d:][path]fname2.ext [offset1 [offset2]]

### Type

External - on CAR: device.

### Availability

As of SDX 4.40.

### Remarks

The program compares both files and displays information about the differences.

```

SpartaDOS X 4.45 | Tue 8-Nov-11 | 15:53:00
D1:DIR
Volume: TESTING
Directory: TESTS
ABCDE    DAT      256 15-02-11 15:46
TEST     DAT       8 15-02-11 15:52
ABZDE    DAT      256 15-02-11 15:46
ABCRAIG  DAT      256 15-02-11 15:46
ANOTHER  DAT      <DIR> 25-01-06 20:19
DATA     ARC     1142 15-02-11 15:46
        688 FREE SECTORS

D1:COMP ABCDE.DAT TEST.DAT
?dif at 000001:00 & 000001:31
?dif at 000002:01 & 000002:32
?dif at 000003:02 & 000003:33
?dif at 000004:03 & 000004:34
?dif at 000005:04 & 000005:35
?dif at 000006:05 & 000006:36
?dif at 000007:06 & 000007:37
?dif at 000008:07 & 000008:9B

D1:█

```

Fig. 19: Compare Given Files

The optional 24-bit offsets count from 0 and are the starting positions, where the respective files get compared from. If none is provided, for both files 0 will be assumed.

If the end of either file is met an EOF will be signaled.

## 4.17 CON - console modes

---

### Purpose

CON: drivers control.

### Syntax

CON 40|64|80

### Type

External - on CAR: device.

### Availability

As of SDX 4.42.

### Remarks

CON.COM is now able to switch the console screen mode between 40, 64 and 80 columns: CON64.SYS<sup>34</sup> and CON.SYS<sup>35</sup> may now be loaded simultaneously and used at will. The side effect is that when the user invokes a driver, which is not loaded (e. g. typing "CON 64" when CON64.SYS is not present in the memory), the screen will return to 40-column mode.

To use all three modes the order of calls in CONFIG.SYS is:

```

DEVICE \DRIVERS\RC_GR8.SYS ; TOOLKIT
DEVICE \DRIVERS\CON.SYS; TOOLKIT
; DEVICE QUICKED
DEVICE CON64
DEVICE RAMDISK 0,8

```

**Notes:** In CONFIG.SYS the DEVICE QUICKED<sup>36</sup> must precede DEVICE CON64 and/or DEVICE CON to get them working properly. DEVICE RAMDISK should be placed behind those in CONFIG.SYS, otherwise you will waste one bank of extended memory.

CON.SYS from the SDXTK needs to be the version released with SDX 4.47 or newer.

CON.SYS is planned to mature into a fullblown hardware independent console driver for both modes, 64 and 80 column text. Until now the 80 column mode is realized.

Please do not mix up CON.SYS with CON.COM from CAR: device.

---

<sup>34</sup> Chapter 8 - Screen Drivers.

<sup>35</sup> Chapter 9 - SDXTK.

<sup>36</sup> Chapter 8 - Screen Drivers.

## 4.18 COPY - copy files

---

### Purpose

The basic task is to copy one or more files to another drive and, optionally, give the copy a different name if specified.

### Syntax

```
COPY [switch] [+][A|H|P] [d:][path][fname].[ext] [d:][path][fname].[ext]/[A]
```

### Type

Internal

### Related

MENU, TYPE

### Remarks

COPY is a versatile tool handling many tasks. It also copies files to the same disk. Doing this, the copies need to be named differently unless different directories are specified; otherwise, the copy is not permitted.

A number of switches are available to customize the execution of COPY to your needs.

Concatenation (combining of files) can be performed during the copy process with the APPEND parameter '/A', which has to be placed at the end of the command sequence.

COPY allows to transfer data between any of the system devices. Some applications of this would be to create a batch file or to print a text file.

COPY will now use free banks of extended RAM (if present) as a part of the copying buffer.

To COPY files from one disk to another having just one drive, the file either has to be COPYed from the source disk to a ramdisk and then from the ramdisk to the destination disk, or, if no ramdisk is available, the MENU program comes in handy as it allows disk swapping.

The first filespec specified is the source. If none is given, a default filename of '\*.\*' is assumed and all files in current directory of the current drive will be copied. However, it is possible to omit the source filespec by using commas to separate parameters.

**COPY, ,C:**

will copy all files from the current drive/directory to the current directory of drive 3.

The second filespec is the destination - if no filename is specified, a default filename of '\*.\*' is assumed (which will copy the files without changing the names).

Remember, if only a filename is specified, the default drive will be used to complete the necessary filespec.

Wildcards ('\*' and '?') are valid in both source and destination filenames. If used in the pathnames, the first directory match will be used.

When using wildcards with the COPY command, the same renaming convention as with the RENAME command applies. The source filespec is used to find directory matches, and the destination filename renames them by overriding characters in the source name with the non-wildcard character in the corresponding position of the destination name.

If a file being copied has no time stamp, current time and date gets assigned to it.

COPY will display a progress indicator when copying files bigger than 64 KiB (unless '/Q' was given in the command line).

These switches are available:

- /B - backup mode
- /C - confirmation mode
- /D - do not preserve date and time
- /I - ask before overwriting a file
- /K - copy and set attribute '+A' to the original file
- /M - delete the source file (move)
- /N - skip existing destination entries
- /Q - do not print anything (except error messages)
- /R - dig recursively into subdirectories
- /S - switch off display during copy
- /V - summary (number of files and directories copied)

The '/B' option enables the program to create backups. It copies all specified entries and applies the +A attribute to mark those files already being saved in a backup. When used next time, it only copies files, which have been created or updated since the last backup. See ATR command for more details on attributes.

The '/C' switch requires confirmation before copying of any file (but not directory, when /R was specified).

The '/D' switch causes a skip of the source file's date and time. The current date and time will be applied to the copy.

The '/I' switch causes the program to check if a file with the same name already exists on the destination. This is done for every file to be copied.

The '/K' switch causes to copy everything normally, but sets the attribute '+A' on the original files. So it is similar to '/B', with the exception that '/B' also skips original files with the attribute '+A' set.

Normally backups are created and updated with '/B', and '/K' is only required, if there is the desire to make a fresh backup instead of updating an old one. In such a case everything is copied, and originals which didn't make it to previous backup updates, are marked '+A'. Once this is done, the next backup update may be done with '/B' again.

The '/M' switch 'moves' files. If source and destination are on different drives, normal copying takes place and then the source file is deleted. To avoid hassles with naming the destination filespec cannot contain a name.



If the source and destination are on the same disk, nothing is physically copied, only the directory entries are moved from the source directory to the destination directory.

Unfortunately, only files can be moved that way. This is the reason, why moving directories is relatively slow - only the directory contents is 'moved' (file by file), whereas the directory itself is re-created at the destination, and deleted at the source place.

`COPY /M` will always clear the archived-bit (+A) on moved files.

Note that the `/M` switch is valid when copying to a character device like `PRN:` and will cause the source file(s) to be deleted.

The `/N` switch causes to skip copying entries (files and directories), which already exist at the given destination.

The `/R` switch allows to copy directories recursively, with all the contents. For example

```
COPY /R A:\ B:\
```

will copy all files and directories (all the contents) from A: to B:, and

```
COPY /R A:>TEST> B:>
```

will copy the contents of the directory `TEST` to the main directory of the disk B:. The directory itself will not be copied.

To copy a single directory with its contents type

```
COPY /R A:\TEST B:\
```

The `/R` switch enables `COPY` to display the source filespecs of the directories being copied.

When copying recursively be cautious and avoid an attempt to copy a directory into itself.

The command sequence:

```
MD TEST
CD TEST
COPY /R >
```

results in 13 nested directories (13, because an attempt to create a further level causes the `COPY` to abort with an error 'Path too long'). Fortunately the command `DELTREE` can delete this.

Recursively copying works also reading MyDOS media. All subdirectories found and their contents will be transferred to the same structure on a SDFS formatted medium.

The `/S` option switches off the screen during the copy process, which will speed up large copy processes remarkably.

When copying from a device other than 'DSK:' (alias 'Dn:' or just 'n:'), just one file will be copied and saved under the destination filespec. For example

```
COPY CON: B:*
```

is illegal because wildcard characters are not allowed in a destination filename, when copying from a character device (or for that matter saving any file).

However, when copying from one character device to another character device, filenames are not used. (Character devices never use filenames.) Example:

```
COPY CON: PRN: .
```

As in the above two examples, when COPYing from 'CON:' the end of file is signaled by pressing <CTRL><3> after typing the text. Also, a <RETURN> must follow each line you enter. That line will be lost otherwise.

Another use for the COPY command is to list files to the printer or screen:

```
COPY README.DOC CON:
```

will display the contents of 'README.DOC' on the screen and

```
COPY README.DOC PRN:
```

will send it to the printer.

Note that both of the above examples could have been performed with the TYPE command:

```
TYPE README.DOC or TYPE README.DOC >>PRN:
```

The COPY command may also be engaged to append files by using a '/A' immediately following (no space) the destination filespec. (SD 3.2 allows a '/A' when SAVEing any file to force append mode - SDX only supports this feature on the COPY command.)

The command

```
COPY NUL: ZERO.DAT
```

is the simplest method of creating a zero-length file.

When a character device (such as CON: or NUL:) has been specified as a source, the switches are treated as follows:

- /I is assumed, unless /N was specified
- /Q is assumed
- /B, /D, /K, /R, /M and /V are ignored

The attributes allowed serve the same function as everywhere in SDX.

The command

```
COPY +H *.* E:\MAKE\TESTDIR\
```

will copy all files being hidden from the current (sub)directory on the current drive to the subdirectory TESTDIR in the directory MAKE on drive E:. The attributes, in this case +H, will not be preserved with the copies of the files.

Keep in mind that COPY will not check, if the filename of a file to be copied already exists on the destination drive. It will overwrite existing files having the same filename, except those files on the destination drive, which have the attributes +H or +P set.

COPY does not preserve file attributes (especially +P).

**Advanced Use:** 'COPY' is a command internal to the Command Processor. The only thing it does, however, is to launch 'CAR:COPY.COM'.

Now it is possible to replace this one with an arbitrary program. The environment variable \$COPY is used to specify, what program to call. E. g.

```
SET COPY=C:>SYS>CP.COM
```

causes the Command Processor to launch the indicated program instead of its defaults and to pass all of the user-specified parameters to it.

## 4.19 DATE - show and set date

---

### Purpose

Display and/or set the current date.

### Syntax

DATE [/T[dd[-mm][-yy]][mm[-dd][-yy]]

### Type

External - on CAR: device.

### Related

CHTD, TD, TIME

### Remarks

Calling the 'DATE' command displays the date in the European (dd-mm-yy) or American format (mm-dd-yy) depending on user selection. The default format is European. The format may be changed using the environment variable \$DAYTIME:

SET DAYTIME=1 - American format

SET DAYTIME=2 - European format

The command 'DATE' produces the following output

```
Current date is 29-09-11
Enter new date (DD-MM-YY):
```

Enter the new date or just press <RETURN> to keep the current setting. The date format - 'DD' for day, 'MM' for month, and 'YY' for year - is obligatory to change the settings. Type 'DD' to change the day, 'DD-MM' to set day and month, all to change the year too. The space key is a legal delimiter.

```
SpartaDOS X 4.46 | Sat 22-Jun-13 | 8:43:49
D1:TIME
Current time is: 20:22:19
Enter new (HH:MM:SS):
D1:TIME 8:42
D1:TIME /T
8:42:53
D1:DATE
Current date is: 25-01-06
Enter new (DD-MM-YY):
D1:DATE 22-6-13
D1:DATE /T
22-06-13
D1:█
```

Fig. 20: Using The DATE Command

When used with the /T parameter only the current date is displayed and no prompt for entering new values will appear.

When fed with a valid date value in the command line, it will set the specified value as current. DAYTIME<sup>37</sup> settings apply here.

Without a proper clock driver<sup>38</sup> installed this command will produce meaningless results. By default, one of the drivers bundled with SDX will be installed during boot up, but can be overridden by creating a custom 'CONFIG.SYS' file to change the preset.

**Notes:** Sometimes a real time clock, when write access is enabled, might get disturbed by programs and then keeps strange data. If this issue cannot be solved using the date and time command or by taking out the battery to reset it, help will come from APETIME.COM<sup>39</sup>. Using a PC connection with APE or AspeQt 0.7.x will download time and date from it and set the respective Atari clock.

AspeQt versions 0.8.x and higher do no longer support this feature. Please use ASPECL.COM instead provided with the AspeQt download package.

RespeQt R3 supports both APETIME and ASPECL.

Additionally, Z.SYS<sup>40</sup> enables to read and set time/date from within a program written e. g. in BASIC.

---

37 Chapter 8 - Important System Variables.

38 Chapter 8 - Time Keeping Drivers.

39 Chapter 9 - SDXTK.

40 Chapter 8 - Time Keeping Drivers - Z.SYS.

## 4.20 DELTREE - delete directory tree

---

### Purpose

Delete subdirectory trees recursively.

### Syntax

DELTREE [/YV] [d:] path

### Type

External - on CAR: device.

### Related

RMDIR

### Availability

As of SDX 4.40.

### Remarks

Before execution, the command asks for confirmation. If permission is granted, it removes the given subdirectory recursively with all the content, successively reporting progress.

The additional switch [/Y] suppresses the confirmation that the program normally needs before executing delete operations. Use it only when you are sure what you are doing.

The /V switch enables a 'verbose' mode, which allows to watch what files are currently being deleted. The pathnames of the directories being deleted and their containing files are displayed.

When error '167 Directory not empty' occurs and the directory seems to be empty, please check the directory to be deleted for hidden files or hidden subdirectories using the DIR command.

If 'Can't delete directory' or 'system error' persists, an invalid directory entry has been found - a file, which was opened for writing, but never closed again (e. g. because of a system crash). Such an entry is invisible in directory listings and cannot be deleted otherwise. The presence of such an invalid entry causes SDX to consider an empty directory as not empty and therefore it cannot be deleted. This error condition indicates a file system structure, which is not completely valid. In this case it is strongly recommended to run the program 'CleanUp X'<sup>41</sup> to verify the structure of the file system and fix it.<sup>42</sup>

---

<sup>41</sup> Chapter 9 - SDXTK.

<sup>42</sup> Appendix E - 'File system consistency checker'.

## 4.21 DEV - kernel devices

---

### Purpose

Display the list of available/installed kernel devices.

### Syntax

DEV

### Type

External - on CAR: device.

### Related

CHKDSK

### Availability

As of SDX 4.43.

### Remarks

It displays four columns: numeric device id, ASCII name of the device, address of the driver, and the information if the I/O being done on the device is buffered by the DOS kernel.

```
SpartaDOS X 4.45 | Tue 8-Nov-11 | 15:53:13
D1:DEV
  Id  Name  Addr  uBufs
  --  ---  -
  0   DSK:  $0E11 On
  1   CLK:  $1261
  2   CAR:  $0B97 On
  3   CON:  $0BB7
  4   PRN:  $0BBA
  5   NUL:  $0C25

  2 slots free
D1:█
```

Fig. 21: Check The Installed System Devices

## 4.22 DF - disk free

### Purpose

Display summary information about free space on all drives.

### Syntax

DF [/A]

### Type

External - on CAR: device.

### Related

CHKDSK

### Availability

As of SDX 4.40.

### Remarks

The command issues a list of all active drives, displaying the total number of KiB available for every single drive letter, the number of free KiB, the percentage of free disk space, and the volume name. An overall summary is displayed at the bottom.

```

SpartaDOS X 4.45 | Tue  8-Nov-11 | 16:00:24
D1:DF /A
Drv  Total      Free  % Volume
===  =====  =====  ==  =====
A:    16128    15704  97  IMPORT
B:     180     172   95  TESTING
C:    16128    15636  96  FILEBOX
D:    139 Device  NAK
E:    139 Device  NAK
F:    139 Device  NAK
G:    139 Device  NAK
H:    139 Device  NAK
I:     128     127  98  X Disk I
J:    138 Device  does not respond
K:    138 Device  does not respond
L:    138 Device  does not respond
M:    138 Device  does not respond
N:    138 Device  does not respond
O:    138 Device  does not respond
===  =====  =====  ==  =====
All  32564    31639  97
D1:█

```

Fig. 22: Display The Properties Of Drives Found

Adding the '/A' switch causes the program to list all the drives from A: to O:, displaying the appropriate error message for unreadable ones. Without '/A' the program lists only drives that allowed the data to be read without errors - the rest is silently skipped.



### 4.23 DIR - directory & DIRS - short directory

#### Purpose

Display a directory in SD or Atari format.

#### Syntax

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

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

#### Type

Both external - on CAR: device.

#### Related

ATR, FIND, MENU, PATH, PAUSE, PROMPT

#### Remarks

'DIR' invoked without any options or switches displays the SD type directory showing filename, extension, file size in bytes, date and time created. A <DIR> in the size field indicates a subdirectory. The volume and directory name will be displayed at the top of the listing, and the free sectors count at the end.

```

SpartaDOS X 4.45 | Tue 8-Nov-11 | 16:03:24
D1:DIR
Volume:      TEST5
Directory:   MAIN
BACKUP      TAR 6988k  16-03-09  21:05
51483      FREE SECTORS
D1:█
  
```

Fig. 23: Display The Directory

Versions before SDX 4.4x display only the last six digits of the file size information in a directory listing, even though the file size can be an 8 digit number. Running SDX such long files can be created and are handled correctly (despite this flaw). Thus, it is rather difficult to properly estimate the size of some long files, but SDX 4.4x solves this problem. When a file exceeds 999,999 bytes in size, it is displayed in kilobytes (KiB), using the 'k' character as indicator.

Reading an AtariDOS 2 type medium is indicated by a volume name of 'DOS 2.0' and a directory name of 'ROOT'; it has no time stamp. The file size is roughly converted to bytes using a multiply of 125 (SD/ED) or 253 (DD), respectively. AtariDOS 2 and clones use sector lengths instead of bytes in the directories, so there is no exact file size representation.

Optionally specify the attributes of the files you wish to display. A '+' with no attribute listed will match all files, regardless of attribute. If you wish to see all files (including hidden files), simply enter

```
DIR +
```

This will also work with any command that allows attribute selection.

The attributes are:

- A Archived files and folders. This attribute is cleared (-) whenever a file is updated or created. Using a program like Flashback! will set this attribute.
- H Hidden files and folders.
- P Protected files and folders.
- S Subdirectory (folder). This attribute cannot be changed.

If you do not specify a filename, '\*.\*' is assumed as in the following examples:

```
DIR MYSUB>
DIR +P
DIR ..\
```

**Notes:** A subdirectory name has to be followed by a '>' or '\' character, if you wish to see the content of that directory. Redirected output will generate listings as for an 80-column display.

The switch '/A' displays the file attributes in the list. It is used mostly together with '+'. For example, to show all files with their attributes use:

```
DIR + /A
```

The '/P' switch causes to wait for a key press after each directory screen (23 lines).

The '/C' parameter will give a count of the number of entries displayed in that directory.

The '/W' switch lists the directory just as file names in as many columns as fit the screen (i. e. in 64-column mode<sup>43</sup>, for example, there are more columns listed than in GRAPHICS 0 with its 40 column size).

You may specify the attributes of the files you wish to display, for example

```
DIR +S
```

will display only subdirectories. Note that the default directory attribute (no attributes specified) is '-H' (do not show hidden files).

The DIRS command has exactly the same syntax, but displays the directory in AtariDOS 2 'compatibility mode' - with no time/date, and with the file size displayed in sectors rather than in bytes. Since the Free Sectors count in DIRS is limited to three digits, the maximum size displayed will always be 999. It also displays the 'protected' status (+P) as '\*' before each protected file and subdirectory.

---

<sup>43</sup> Chapter 8 - CON64.SYS driver.

## 4.24 DUMP - display file contents

### Purpose

Display a file in HEX and ATASCII form.

### Syntax

DUMP [d:][path]fname[.ext] [[\$]start] [[\$]len] [/A]

### Type

External - on CAR: device.

### Related

TYPE

### Remarks

The parameters 'start' and 'len' are the start addresses in the file and the number of bytes to dump (respectively). They are assumed to be in decimal format unless preceded by a '\$' (in which case they are HEX format).

The '/A' switch is added to cause some ATASCII specific characters (semigraphics, inverse video characters etc.) to be replaced with dots. This allows to print the DUMP output on a printer, especially, if the printer interface does not allow full code translation or if it's not using a graphics mode to print.

DUMP is useful to quickly examine the content of a file. To modify a file or examine and modify disk sectors, use 'Eddy'<sup>44</sup>.

```

C:\DUMP CAR:EXIT.COM
0000- FE FF 01 00 00 00 25 00
0008- AD F9 FF C9 FF D0 01 60
0018- E0 00 D0 04 2C A9 00 2C
0020- A9 9C 48 20 00 00 20 00
0028- 00 68 4C 00 00 FC FF 01
0030- 00 00 40 45 58 49 54 20
0038- 20 20 FB FF 43 4F 4D 54
0040- 41 42 20 20 00 00 FE 01
0048- 01 FC FB FF 46 49 4C 45
0050- 4C 45 4E 47 00 00 FE 01
0058- 1C FC FB FF 46 53 45 45
0060- 4B 20 20 20 00 00 FE 01
0068- 1F FC FB FF 55 5F 46 41
0070- 49 4C 20 20 00 00 FE 01
0078- 23 FC FB FF 55 5F 47 45
0080- 54 4E 55 4D 00 00 FE 01
0088- 0C FC
C:\

```

Fig. 24: File Dumped To The Screen

**Note:** Invoked without any parameter the system will wait for an input from the CON: device. Pressing <CTRL><3>, <BREAK> or <RESET> aborts.

## 4.25 ECHO - echo command line input

---

### Purpose

'Echo' command line input.

### Syntax

ECHO ON|OFF

ECHO [/N] message

### Type

Internal - executed by COMMAND.COM

### Availability

As of SDX 4.40.

### Remarks

It echoes command line input passed to the Command Processor from the command line or fetched from a batch file. The default is ECHO OFF.

When enabled it serves two functions:

1. The text given as a parameter is simply displayed on screen.

ECHO TEXT        echoes the word TEXT.

2. Display the value of a environment variable.

ECHO \$PATH       displays the current path setting.

ECHO used in conjunction with output redirection<sup>45</sup> is a helpful tool.

The /N switch is only taken into account when displaying text. When /N is given, the ending EOL (which is added by default) gets suppressed.

---

<sup>45</sup> See Chapter 5 – I/O redirection.

## 4.26 ED - editor

---

### Purpose

Enable text editor.

### Syntax

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

### Type

External - on CAR: device.

### Availability

As of SDX 4.40.

### Remarks

ED.COM is a SDX compliant, relocatable version of JBW Edit. The main purpose of the program is to edit the DOS configuration files, but it obviously can be used to edit any text files. The practical file size limit is about 6-8 KiB, even if the editor buffer is much larger.

The default height of the editor's window is 10 lines. The declaration of the environment variable \$ED can change that. E. g. SET ED=20 will set the height to 20 lines. Values from 1 to 22 are allowed. Exceeding this range causes the ED to assume the maximum possible size (i. e. 22 lines).

Calling ED with a filename on the command line causes an attempt to load the named file. The editor works in auto insert mode - typing a character inserts it at the current cursor position and moves everything to the right.

Available editing commands:

<ESC> Cancel the function or quit the program.

<CTRL><L> Load - load a file into the buffer.

<CTRL><S> Save - save the buffer to a file.

<CTRL><U> Up - move the low margin of the editor window up.

<CTRL><D> Down - move the low margin of the editor window down.

<CTRL><V> Visible - make EOL characters visible.

<F1> or <CTRL><up arrow> Cursor up

<F2> or <CTRL><down arrow> Cursor down

<F3> or <CTRL><left arrow> Cursor left

<F4> or <CTRL><right arrow> Cursor right

<CTRL><B> Begin - move the cursor to the beginning of the text.

<CTRL><E> End - move the cursor to the end of the text.

<CTRL><A> Move the cursor to the beginning of the line.

<CTRL><Z> Move the cursor to the end of the line.

<CTRL><T> Tag - tag the current line.

<CTRL><G> Go - move the cursor to the tagged line.

**<CTRL><Q>** Quit - quit the control mode, the next key combination will be interpreted as a character.

**<SHIFT><CTRL><up arrow>** Page up.  
**<SHIFT><CTRL><down arrow>** Page down.

**<CTRL><INSERT>** Input a space at the current cursor position and move everything to the right. Cursor keeps the position.

**<CTRL><DELETE>** Delete the character under the cursor and move everything to the left. Cursor keeps the position.

**<SHIFT><INSERT>** Inserts a new line (which is a copy from that line where the cursor is standing) before the tagged one (see **<CTRL><T>**), and moves the cursor to the next line.

Pressing this key combination several times in a row allows to quickly copy a consecutive number of lines to another section, e. g. in a batch file.

**<SHIFT><DELETE>** Delete the current line and move the rest up.  
**<DELETE>** Delete the character under the cursor and move everything to the left.

**<SHIFT><CTRL><E>** Erase - clear the editing buffer.

**Note:** Currently ED.COM works only in GRAPHICS 0. Wildcards are not allowed for saving new files! Do not edit text files, which contain paragraphs in length of more than 128 characters!

```

SpartaDOS X 4.45 | Tue 8-Nov-11 | 16:05:36
Edit DA:AUTOEXEC.BAT
POKE 82,0
POKE 710,6
POKE 709,14
SET SCRDEF=14,6
SET MANPATH=A:\HELPSYS
SET PAGER=LESS;/C;%
PROMPT $L$P\
DATE
TIME
Free:30187-Line:1 Col:1 80
D1:ED AUTOEXEC.BAT

```

Fig. 25: Small Text Editor

## 4.27 ERASE - erase files

---

### Purpose

Delete the file in the specified directory on the designated drive, or delete the file from the default drive if no drive is specified. If no path is specified, the file is deleted from the current directory.

### Syntax

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

### Alias

DEL & DELETE

### Type

Internal

### Related

MENU, UNERASE

### Remarks

You can use wildcards such as '\*' and '?' to delete multiple files, but use caution since usually no warning is given. If you do enter \*.\* as the specifier, SDX will prompt you with:

Erase ALL: Are you sure?

With SDX 4.4x additional precautionary checks have been implemented to make sure that inputs like '?\*.\*' will not accidentally kill your data. Use the menu<sup>46</sup> for tagging files to delete.

---

46 Chapter 4 - Menu Program.

## 4.28 FIND - find files

---

### Purpose

Search specified areas in the system for files.

### Syntax

```
FIND [d: | device]fname[.ext]
```

### Type

External - on CAR: device.

### Related

DIR

### Remarks

FIND will quickly find a file anywhere on your drives. This becomes very useful when you start using subdirectories and multiple drives.

```
SpartaDOS X 4.45 | Tue 8-Nov-11 | 16:05:55
D1: FIND *.DAT
D1: > SP_SUPER.DAT

D2: > TESTS > ABCDE.DAT
D2: > TESTS > TEST.DAT
D2: > TESTS > ABZDE.DAT
D2: > TESTS > ABCRAIG.DAT

D3: > TEST > ZERO.DAT
D3: > TEST > 00.DAT

 7 matches found!
D1: █
```

Fig. 26: Find Files On The System Drives

Entering drive id and filename will tell FIND to look only on that particular drive. Additionally, FIND is able to search devices like 'CAR:', if specified.

The filename may include wildcards. All filename matches found will be displayed with the full path from the root directory to the filename match. The drive ids will be shown as drive letter for consistency with drives above D9:. The number of matches found will be displayed at the end of the search. FIND will also find and display hidden files.

FIND was inspired by WHEREIS.COM from legacy SDTK.



## 4.29 FMT - format text

---

### Purpose

Simple text formatter.

### Syntax

FMT [/S/J] [<<][path]fname[.ext] [ncol]

### Type

External - on CAR: device.

### Related

MAN, MORE, LESS, TYPE

### Availability

As of SDX 4.42.

### Remarks

A simple text formatter, which reads the input line by line and applies to each one:

- Spaces at the beginning of the line are removed.
- '/S' switch: Multiple spaces are turned into one single space.
- If the resulting line is longer than 'ncol' characters, it will be folded.
- '/J' switch: The line gets justified to both margins, unless an EOL character is at the end.

The 'ncol' value may not be less than 32 or more than 127. When 'ncol' was not specified, the current screen width is assumed.

The line folding and justification works best, when the text is prepared so that single paragraphs (no matter how long) are terminated by one or two EOL characters, but do not contain EOLs themselves (continuous text). To prepare such a file a text editor can be used that can do proper line wrapping and does not enforce terminating each line with the <RETURN> key.<sup>47</sup>

FMT is a filter command that can receive data from a pipe<sup>48</sup>. Its primary purpose is helping to format documentation files for the MAN command. For example, if you want to print one of the help files (say HELP.DOC) on paper in 80 columns and justified form, do this:

```
MAN HELP /P | FMT /S /J - 80 >>PRN:
```

If the text was not prepared according to the remarks above, it is a good idea to preview the FMT output on the screen first.

**Note:** Invoked without any parameter the system will wait for an input from the CON: device. Pressing <CTRL><3>, <BREAK> or <RESET> aborts.

---

<sup>47</sup> e. g. the 1<sup>st</sup> XLEnt Wordprocessor.

<sup>48</sup> Chapter 5 - Pipes.

## 4.30 FORMAT- format disk

---

### Purpose

Initialize a disk in either SD or AtariDOS 2 format. You may select density, sector skew, tracks, and volume name before formatting.

### Syntax

FORMAT [/Q dn: [volname]]

### Type

External - on CAR: device.

### Related

BOOT, CHVOL

### Remarks

The formatter is capable of handling any Atari compatible media. As of SDX 4.47 FORMAT is now an utility residing on the CAR: device. Executed without arguments it invokes the formatter menu as usual. Adding a /Q switch followed by a drive id and an optional 8-character volume name will soft-format the specified drive without a need to enter the formatter menu. If no volume name is provided, it will be set to 'NO NAME'. Soft-formatting will convert any readable medium to SDFS 2.1, no matter what the current file system was before formatting.

The formatter is a menu driven program, that allows you to initialize just about any type of medium that works with an A8 computer when called from the command processor by typing FORMAT. From within a program called by XIO 254<sup>49</sup> allows FORMAT to be used with most programs that support disk initialization.

Formatting a storage medium consists of several steps. At first the sector structure is written to it. Now the DOS has a place to put the program information to. Next the directory structure is written to it, wherein the DOS keeps track of the sector usage. Also ramdisks, partitions on hard disk drives or memory cards or partition-like media (e. g. ATR image) can be initialized using the FORMAT menu, but only using the BUILD DIRECTORY option, since it is assumed that the medium is already low level formatted in the right way. The formatter provides some automated assistance as it tries to read information from the addressed drive or partition about type, density or volume to identify it most accurate.

```

SpartaDOS Formatter <esc>=exit
Unit #:  █          Volume:
Skew: HSpeed  Density: Double
Mode: Sparta  Tracks: 40 55
Optimize: Off          720 Sectors
          256 BP5      184320 Bytes
Specify a drive to continue...

```

Fig. 27: Formatter Menu

Exit the FORMAT menu at any time by pressing <ESC> before formatting begins.

After entering the FORMAT menu choose the following parameters:

- U **Unit** is the initial selection that tells the formatter which drive you wish to initialize. Valid choices are: 1 - 9 or A - O. After entering the unit number or letter the program reads the drive to determine what type it is. The formatter automatically determines whether the drive is a serial drive and if it is configurable or if the drive is a ramdisk or hard drive, which appear the same at this point. When the selected drive is identified as a ramdisk or a hard disk partition, SDX 4.4x attempts to read the existing volume name and presents it in the formatter menu as the default one.
- O **Optimize**. Previous versions of SD build directories marking the last sector on the disk as occupied and leaves it unused<sup>50</sup>. With Optimize enabled that sector will be reclaimed and assigned to the data area, providing one more free sector on freshly formatted media.
- S **Skew**<sup>51</sup> refers to the order in which the sectors are arranged in a track. The three valid choices are: Ultra Speed, High Speed and Standard. High Speed will automatically put the correct Ultra Speed skew on a disk with the 1050 US Doubler or Indus GT drives. It will also put the correct high speed skew on the Atari XF551 drive under Double Density. Standard skew is used on all other floppy disk drives. When the drive does not support a selected high speed mode, the format program will receive an error from the drive and then try to format in Standard skew. For the fastest possible reading and writing, on most drives the correct skew is required. Skew is not applicable to ramdisks, and hard drives, since they can not be physically formatted by this program.
- M **Mode** is either **Sparta** for the SDFS directory structure or **Atari** for all the AtariDOS 2.0 clones and their directory structures in single and double density only. If you switch from Sparta to Atari mode, impossible format settings will automatically be re-adjusted to legal Atari settings.
- V **Volume** is a way of naming the media for organizational purposes. Up to eight characters are allowed on SDFS formatted media. The volume name may contain any ATASCII characters except spaces. Volume is used on SDFS media only and is not applicable to other DOS types.
- D **Density** offers the choice of the five types used with A8 computers as there are:
  - Single, 128 bytes per sector FM,
  - Medium, 128 bytes per sector MFM (also known as Dual or Enhanced),
  - Double, 256 bytes per sector MFM,
  - DD 512, 512 bytes per sector MFM,
  - High, high density, enabling to use 1.440 KiB floppy disk drives.

Stock Atari 810s only support single, upgraded 810s (e. g. Happy Enhancement) double density.

Stock Atari 1050s support single and medium density, enhance ones (e. g. Speedy), Atari XF551s, and Indus GTs support single, medium, and double density. Most other disk drives for the A8 support single and double density.

---

<sup>50</sup> Appendix H - Optimize for users of ancient hard drives.

<sup>51</sup> Appendix H - Skew for more details.

DD 512 is supported by certain types of hard drive interfaces (KMK/JŽ IDE 2.0 Plus, IDEa, and IDE Plus), with TOMS floppy disk drives (TOMS 710, TOMS 720, or third party drives with TOMS Turbo Drive or TOMS Multi Drive extensions installed). With DD 512 selected SDX 4.4x forces the Sparta mode as there is no possibility to build an AtariDOS file system for the disk with 512-byte sectors.

High density is supported e. g. by the HDI<sup>52</sup>.

**Note:** Medium density is now available for 1050 drives, but SDFS format only. Additionally, non-PERCOM drives (Atari 810, 815, 1050) are handled properly now.

T **Tracks** can be **40 SS**, **40 DS**, **77 SS**, **77 DS**, **80 SS**, and **80 DS**.

SS means Single Sided (1 head writes on one side of the disk) and DS means Double Sided (2 heads with each writing on opposite sides of the disk). All Atari brand drives will use 40 SS except for the XF551, which is capable of 40 DS. Most of the other 5.25" drives will be either 40 SS or 40 DS. 77 Tracks is used for 8 inch disk drives connected via an interface like the ATR8000 or PERCOM controller. 80 Tracks is used for 3.5" drives and high capacity 5.25" drives with a similar interface. All drives with two heads will also format in the SS mode.<sup>53</sup>

**Note:** The drive controllers do not provide adequate feedback to the computer when formatting a disk/medium to determine whether the tracks selection is wrong for the drive. It is important to enter the correct information, otherwise the disk/medium will end up with an incorrect free sectors count.

F **Format Disk** will start the physical format of a floppy disk assuming all required parameters are set. The physical format and verify of it are functions of the floppy disk drive controller. Next the directory structure selected in Mode is written to the disk and verified. CAUTION: The Format Disk procedure obviously destroys all previous stored information on the disk.

**Note:** Drives recognized as ramdisks or hard drive partitions will be soft-formatted, meaning only their directories will be written anew. A ramdisk must be installed with the RAMDISK.SYS driver.<sup>54</sup>

B **Build Directory** is the initialization option available for ramdisks and hard disks, although it will work equally well with floppy disk drives. The only parameters available for these disks are Unit number and Volume name. The others are predetermined or not applicable. Build Directory writes a fresh SD directory structure to the drive unit selected, which means it will destroy access to all previous information stored there.

The physical format of a hard disk drive must be performed by a special program written for the particular hard drive, interface, and controller. That is considered a low-level format and is beyond the scope of the formatter menu. Same applies to modern storage media like CF cards, etc. The physical format of a ramdisk is provided by the ramdisk handler at installation.

---

<sup>52</sup> Appendix H - High Density Interface.

<sup>53</sup> See your disk drive manual for details.

<sup>54</sup> For 65C816 system see the SDXTK for a proper driver.

Sectors and bytes counts shown on the FORMAT menu are determined by what is read from the configuration of ramdisks or hard disks, or by the parameters selected for a floppy disk drive format.

As of SDX 4.42 the formatter will verify, if the drive has selected the correct parameters, and it will ask for confirmation in case of incongruity. Unfortunately, only 'Standard' and 'High Speed' formatting protocols allow to detect the error before actual formatting. In 'UltraSpeed' drives the disk must be formatted first, and then it is possible to check, if the disk's capacity is the same as expected.

When in doubt, you can use the XFCNF command from the SDXTK to check, if the particular density is correct for the drive you use.

**Indus GT notes:** The Indus GT before version 1.20 has a few quirks. Because of this, medium density is not supported on this drive for the Atari file system, but will work with the SD file system. The Indus GT has also been known to keep spinning indefinitely when used in a system with US Doubler enhanced 1050s. If this problem occurs, the best solution is to stop using mixed drives in one system.

**CAUTION:** The formatter uses the 6502 stack space intensively. Because of that, some floppy disk drive turbo systems, which load the fast serial I/O patch onto the stack, will not work in turbo mode. Problems will occur with Top Drive 1050, TOMS Turbo Drive or any other using the same method. Such a drive can be used with SDX, but only at the standard baud rate.

Other speeder systems, such as TOMS Multi Drive (in the Ultra Speed mode), TOMS 710/720, CA-2001, Atari XF551, LDW Super 2000, Indus GT, Happy, 1050 US Doubler will work normally.

**Other Notes:** More high speed disk drives tested successfully are XF551 with HyperXF ROM, HDI, 1050 Speedy (all versions).

Rana 1000, TRAK-AT and 1050 Turbo have successfully been used in normal mode.

When the VBXE driver from the SDXTK is loaded, the formatter will use it to temporarily disable the 80-column-display, so that it does not interfere with the formatter menu.

An existing volume name is displayed only for ramdisks, hard drive partitions, or ATR files, which identify themselves as such. If an ATR file identifies itself as floppy, the existing volume name (or anything at all) is not read from the boot sector, nor displayed. This is done to avoid a situation, when the formatter gets blocked for a longer time at the beginning, because there is no floppy in the drive.

Booting a medium formatted by SDX having no bootable program causes the message "ERROR - No DOS" to be displayed.

### 4.31 KEY - keyboard buffer

---

#### Purpose

Installs a 32 character keyboard buffer.

#### Syntax

KEY ON|OFF

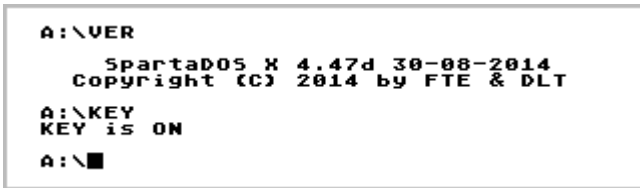
#### Type

External - on CAR: device.

#### Remarks

The first time you use this command, it installs a keyboard driver into your system and links an 'internal' KEY command into your system (for turning the buffer on and off). The keyboard buffer will provide a faster key repeat and enables you to type ahead while the system is busy.

The ON/OFF parameter is interpreted, enabling or disabling the keyboard buffer accordingly while reporting the ON/OFF status whenever invoked. Keys coming from the auto-repeat of the keyboard will not be buffered.



```

A:\VER
      SpartaDOS X 4.47d 30-08-2014
      Copyright (C) 2014 by FTE & DLT
A:\KEY
KEY is ON
A:\█
  
```

Fig. 28: KEY Buffer Is ON

Once the keyboard buffer has been installed, the global symbol '@KEY' is defined and further KEY commands call this symbol to turn the buffer on and off.

**Notes:** The keyboard buffer may be incompatible with some programs but is more compatible with other programs than the SD 3.2 buffer (most notably with the ACTION! cartridge).

KEY now handles key combinations <CTRL><F1-F4>. The 1200XL panel LEDs are not switched to avoid a potential conflict with memory expansions utilizing port b of PIA.

Please set KEY to OFF when flashing SDX ROMs manually.

### 4.32 LESS - view text

---

#### Purpose

Paging text viewer.

#### Syntax

LESS [/C] [<<][path]fname[.ext] [>>][path]fname[.ext]

#### Type

External - on CAR: device.

#### Related

MORE, TYPE

#### Availability

As of SDX 4.42.

#### Remarks

LESS is a advanced version of MORE. If the text fits entirely on the screen, the viewer behaves almost like MORE or TYPE: it dumps the file's content to the screen, and then quits to the DOS. The only difference between LESS and the others is that LESS tries to fold the long lines (if any) so that they fit within the current screen width.

Refer to the instructions on the FMT command for further remarks about folding.

However, if the text is longer than 23 lines, it will be 'paged' and the viewer will not automatically terminate. The following keystrokes are available for navigation through the viewed file:

- <Down arrow>, <F2> or <RETURN>: scroll one line down
- <Up arrow> or <F1>: scroll one line up
- <Right arrow>, <F4> or <F> or <SPACE>: scroll one page down (forwards)
- <Left arrow>, <F3> or <B>: scroll one page up (backwards)
- <D>: scroll half page down
- <U>: scroll half page up
- <G>: jump to line number of the text
- <<>: jump to the top of the text
- <>>: jump to the end of the text
- <Q>: or <ESC>: exit to DOS

The keys <- , =, +, \*> are equivalents to the respective arrows and do not require holding the <CTRL> key.

Some of the keys are shown in the bottom of the screen. The number in the bottom-right corner is the number of the text line displayed at the top of the screen.

Adding the '/C' switch causes the program to clear the screen before displaying anything. LESS is also a filter command and thus can be used in an identical manner as MORE. It can act as the final receiver of the data stream sent through a pipe<sup>55</sup>, i. e.:

D1:DIR | LESS

```

SpartaDOS X 4.45 | Tue 8-Nov-11 | 16:13:48
Volume:  IMPORT
Directory: MAIN
AUTOEXEC  BAT      115  25-01-06  20:17
ARCLOCK3  SYS      609  9-01-11  18:34
ARCLOCK4  SYS      688  9-01-11  19:31
PAGEMAP   COM      737
PAGEFIND  BAS     5299
XRAM021   COM     2438  29-10-88  14:00
MEM0      COM      224  29-12-08  11:20
MEM2      COM      225  29-12-08  11:21
MEM3      COM      238  29-12-08  11:21
TURBOBAS  COM    18104  9-02-92  14:21
JACFLD95  HEX    24299  23-01-11  20:18
FASTMIX   HEX    40029  23-01-11  20:18
FIXED     HEX    39970  23-01-11  20:18
SPARTA    DOS  <DIR>   9-01-11  20:50
PARTY     HEX    39882  23-01-11  20:18
ARCLOCK5  SYS      688  24-01-11  10:05
COMPILER  COM     9933  9-02-92  14:25
RUNTIME   COM    10876  9-02-92  14:26
8PLAYDEM  XEX    24045  26-01-11  16:32
[2] line [3] page [4] ends [5] quit 0

```

Fig. 29: Less Used To Format A Directory Output

LESS now automatically detects MS-DOS, CP/M and Unix special characters and converts the text accordingly on the fly. Therefore, you now may use the command to convert such text files to Atari format, in this manner: LESS FOO.TXT >>BAR.TXT will convert a PC-like text file FOO.TXT into Atari-like BAR.TXT. The mapping table for special characters to be converted:

ASCII 7 BELL	→	ATASCII 253 BELL
ASCII 8 BACKSPACE	→	ATASCII 126 BACKSPACE
ASCII 9 TAB	→	ATASCII 127 TAB
ASCII 10 CR	→	ATASCII 155 EOL
ASCII 13 LF	→	ATASCII 27 ESCAPE
ASCII 127 DELETE	→	ATASCII 126 BACKSPACE
ASCII 160 HARD SPACE	→	ATASCII 32 SPACE-separated

Line ending characters, which can be CR, LF or CR/LF, are basically converted to EOLs (ATASCII 155).

LESS, when redirecting its output to a file, will now skip text folding altogether (just the above character conversion takes place, if applicable).

**Notes:** LESS loads the file to be displayed into free main memory and reports to be 'Out of RAM', if it does not fit.

Invoked without any parameter the system will wait for an input from the CON: device. Pressing <CTRL><3> and then <RETURN>, <BREAK> or <RESET> aborts.

The intention for LESS is to display a formatted output of text files. Trying to display other file types may deliver unexpected results.



### 4.33 LOAD - un-/load a file

---

#### Purpose

Load/unload a file (no run) to/from memory.

#### Syntax

LOAD [/X|L S|A|N|I][mode] [d:][path][fname][.ext] [parameters]

#### Type

Internal

#### Related

MEM, SAVE

#### Remarks

LOAD may be used to:

- Keep an external command such as CAR or X or the command processor (COMMAND.COM) resident in memory.<sup>56</sup>
- Remove all non-installed commands or programs from memory (use LOAD with no filename).
- Load a subprogram into memory for use by other commands.
- Load MAC/65 object files into memory and then SAVE them back as contiguous non-segmented binary files.
- Load a binary program prior to running a debugger (for testing purposes).

Via switches special conditions can be invoked:

- /X will execute the file under the control of X.COM (equivalent to the command 'X fname').
- /L allows advanced users to select a loading mode by symbol or mode value:

Symbol	Mode	Loading Mode
S	0	SDX loading mode
A	64	AtariDOS loading mode
N	128	No execution
I	192	Initialize only, no execution.

If not specified, the loading mode with the LOAD Command is assumed to be N (128).

**Note:** Useful for keeping commonly used commands resident in memory, thereby eliminating the need for these commands to load from disk.

---

<sup>56</sup> Chapter 4 - MEM for more details.

## 4.34 MAN - display online documents

---

### Purpose

Starts the documentation viewer.

### Syntax

MAN [fname] [/P?]

### Type

External - on CAR: device.

### Availability

As of SDX 4.40.

### Remarks

MAN is a basic text viewer, whose usage is customizable to your needs. Its main purpose is to view documentation files. The program's operation is similar to the man command known in UNIX, where it is an abbreviation for manual, thus the name.

As a prerequisite the environment variable \$MANPATH must be defined. Type

```
MAN
```

to see all available man files. If MANPATH is not defined yet, you will see a respective notice. Otherwise the available man files will be displayed in the order of the directories given to MANPATH.

```

SpartaDOS X 4.45 | Tue 8-Nov-11 | 11:28:07
Copyright (C) 2011 by FTE & DLT

A:\MAN
Available help files:

APPEND    ARC        ATR        BASIC
BLOAD    BOOT      CAR        CHDIR
CHKDSK   CHTD     CHVOL     CLR
CLS      COLD     COMMAND   COMP
CON      COPY     DATE      DELTREE
DEV      DF       DIR       DUMP
ECHO     ED       ERASE     FIND
FMT      FORMAT   KEY       LESS
LOAD    MAN      MAP       MDUMP
MEM      MENU     MKDIR     MORE
PATH    PAUSE    PEEK      POKE
PROMPT  PWD      RENAME    RENDIR
RMDIR   RS232   SAVE      SET
SETPATH5 SIOSET  SORTDIR   SWAP
TD      TIME     TYPE      UNERASE
VER     VERIFY   X

A:\

```

Fig. 30: Available Man Pages With SDX

The MANPATH contains a list of directories to be searched for text files, and may be set in a batch or config file. Example:

```
SET MANPATH=C:>MAN;D:>DOC
```

Put a file in any of these directories with a \*.HLP, \*.MAN, \*.DOC or \*.TXT extension, and hand its name (but omitting the extension) over to MAN. The file will be searched for

and displayed when found. If something went wrong in defining MANPATH you will see the message again.

```
Type      MAN MAN
```

or even more convenient

```
MAN MAN | LESS
```

to read about how to use MAN.

To check if the desired man page is available in the current MANPATH type

```
MAN fname /?
```

MAN, when multiple help pages with the same name are found along the \$MANPATH, displays a list requesting to pick up one.

MAN is able to automatically unpack a \*.MAN file if it has been compressed with ARC. Simply pack your man files using ARC, each one separately, then rename the resulting archives from \*.ARC to \*.MAN.

**Note:** Unpacking on the fly requires a temporary file, so better define your \$TEMP to point to a fast (parallel) hard disk or a RAM-disk.

An example for organizing and addressing files: the HDSC.ARC archive contains a text file named HDSC.DOC filled with information about the program. Having unpacked the archive, you can put the executable to any directory pointed to by PATH, and the \*.DOC file to any of the directories pointed to by MANPATH. If you want to read the instructions, you do not need to remember where all the files are. Just type

```
MAN HDSC
```

and HDSC.DOC appears on the screen. If the text file has 'long lines', or in an extreme case the entire file consists of a single line, the viewer will try to justify the text so that it fits to the current screen width.

MAN just dumps the content of the file to the screen without any processing. When the file ends, it quits to DOS. The display does not get paged and the program does not ask the user, whether to proceed or not - it works just like TYPE, when the /P switch was omitted. This is useful, when you want to redirect MAN's output to another file or device, e. g. a printer. To make a hard copy of the file HELP.DOC, you just need to type :

```
MAN HELP >>PRN:
```

**Note:** See the FMT command description for additional formatting methods available.

Apart from usual ways like **MAN HELP | MORE** an external viewer can be registered for MAN.COM to facilitate advanced paging functions: TYPE (with the /P switch), MORE, and LESS.

To register a viewer, just assign its name to the environment variable \$PAGER.

For MORE and LESS, this is done with

```
SET PAGER=MORE or SET PAGER=LESS
```

The alternative, 'TYPE frame.ext /P' acts exactly like MORE (or vice versa), so it is impractical to use it instead of MORE. We supply the method of registering it, because it can serve as an example on how to register an external viewer that needs additional parameters.

In order to have an external viewer work, the MAN must first construct a command line out of the template stored in the \$PAGER, the MANPATH and the filename found among the help files. Since the /P switch must be given to TYPE at the end of the command line, there has to be a method for telling MAN where the filespec has to be inserted. Next, the filespec must be separated from the rest of the parameters with spaces. But, an environment variable (as \$PAGER) may not contain spaces.

So, in the command line template the filespec's place is marked with the '%' character (when you omit it, the filespec will be simply appended at the end), and spaces are replaced with semicolons. Thus the command to register TYPE /P as an external viewer for MAN looks like this:

```
SET PAGER=TYPE;%;/P
```

This is useful when the pager you use requires (or allows) some options to be selected. For instance, if you want to use LESS as the pager, and you want it to clear the screen before displaying the help file, do this:

```
SET PAGER=LESS;/C;%
```

The '/P' switch for MAN causes it to ignore the external viewer and use the default method (i. e. TYPE).

**Notes:** Sets of man pages for commands and drivers of SDX are available as part of the SDXTK. Additionally, builds for several hardware platforms already contain the man pages for commands. \$MANPATH defaults to 'CAR:' then. If you change MANPATH, keep 'CAR:' in it for the man pages stored in the cartridge.

Additionally, the programs from SDXTK and SDX Add-Ons come with a respective DOC, TXT or MAN file.

### 4.35 MAP - map drives

---

#### Purpose

SIO.SYS control.

#### Syntax

MAP [unit] [SIO|OS|NORMAL|OFF] [d:] or MAP [d:]fname[.ext]

#### Type

External - on CAR: device.

#### Availability

As of SDX 4.40.

#### Remarks

MAP invoked without any parameter displays a help screen. The command 'maps' the given drive id 'd:' to the drive associated with the specified number (unit). An internal SIO translation table is used here. The additional options control the communication mode for the specified disk:

- NORMAL - standard mode, PBI has priority over SIO.
- SIO - SIO communications only (PBI is bypassed).
- OS - communication is redirected to the ROM OS.
- OFF - drive disabled (or handled by another driver).

The 'SIO' option allows to gain access to a serial drive, which has been masked out by a parallel (PBI) drive having the same number. For instance, the command 'MAP 1 SIO B:' creates a logical drive B: (or D2:), which uses physical disk drive number 1. 'MAP 3 K:' creates a logical drive K: (or DK:), which uses drive number 3. Drive #3 can be a SIO or a PBI drive. 'MAP 11 K:' will remap drive #11 to K.

The 'OS' parameter applies, when the OS ROM routines are to be used instead of the native SD communication routines. 'OS' cannot be used, when the DOS is configured to USE OSRAM mode.

MAP [d:]fname[.ext] calls SIO parameters to be changed from a text file, whose name can be handed over to the MAP command as the only argument. Each line of that text file has to contain correct MAP parameters starting from the unit value. This feature allows for changing the settings of multiple drives at once.

**CAUTION:** MAP does not work, when SIO.SYS is installed using the '/A' or '/C' option!

**Note:** MAP works in addition to the SWAP command and drive remapping set by MIO, Blackbox, MSC or alike. So take that into consideration when using such hardware. It is very easy to lose track of which floppy disk drive, ramdisk, or hard drive partition is at what logical drive.

## 4.36 MDUMP - display memory contents

### Purpose

Display memory contents in hex and ATASCII.

### Syntax

MDUMP symbol | [[ $\$$ ]index:][ $\$$ ]address<sup>57</sup> [[ $\$$ ]len] /R

### Type

External - on CAR: device.

### Related

DUMP

### Availability

As of SDX 4.40.

As of SDX 4.47 can access extended memory and 65C816 high RAM.

### Remarks

MDUMP is useful to check the memory content quickly.

```

SpartaDOS X 4.47 | Mon 9-Dec-13 | 11:25:48
A:\MDUMP $D300 8
  D300- FF FF 3C 3C FF FF 3C 3C  █<<<|>>>
A:\MDUMP 54016 8
  D300- FF FF 3C 3C FF FF 3C 3C  █<<<|>>>
A:\MDUMP 54016 $8
  D300- FF FF 3C 3C FF FF 3C 3C  █<<<|>>>
A:\MDUMP $FF0000 $8
$FF:
 0000- FF FF FF FF FF FF FF FF  ██████████
A:\MDUMP 16711680 8
$FF:
 0000- FF FF FF FF FF FF FF FF  ██████████
A:\█

```

Fig. 31: Memory Dump

Available indices are   00 = main memory,  
                           02 = system extended memory,  
                           04 - max. 67 (255) = extended memory Port B (Axlon).

The switch '/R>' will cause addresses to be displayed as offsets relative to the given starting address.

To dump 65C816 high ram type the respective number of digits, e. g. \$FF0000.

**Note:** Addresses are assumed to be decimal unless preceded by a '\$', which indicates hex. Hex and dec values may be mixed.

<sup>57</sup> See Chapter 6.4 for new functionality in calculating an address.

### 4.37 MEM - display memory configuration

#### Purpose

Display the current memory usage information.

#### Syntax

MEM [/X]

#### Type

External - on CAR: device.

#### Related

CHKDSK, LOAD, MDUMP

#### Availability

As of SDX 4.47 indicates 65C816 high memory.

#### Remarks

This command uses the program MEM.COM on CAR: device. It displays all information about the current memory configuration. The mode it is being used by DOS (NONE, OSRAM, BANKED), the bottom limit of available user and extended memory, and at last how many unused memory banks are available. Two limits for each memory region are given - the first being the top limit of installed drivers and the second being the top limit of held-in-memory applications.

```

SpartaDOS X 4.46 | Tue 18-Jun-13 | 18:51:48
C:>MEM
Main: $1422,$1440
Ext: $708D,$7687
Use: BANKED

4 banks free
C:>■

```

Fig. 32: Memory Information

In the example shown here the installed drivers use memory from \$700-\$1421 and \$4000-\$708C, and the used applications held in memory reside from \$1422-\$143F and \$708D-\$7686.

The applications held in-memory are LOADED into memory and consist of files such as COMMAND.COM and X.COM<sup>58</sup>. The drivers installed in memory are files such as SPARTA.SYS, ATARIDOS.SYS, RAMDISK.SYS, etc.<sup>59</sup>

Normally the first and second numbers will be the same. The OS variable MEMLO (at \$2E7) contains the second number. If LOAD is executed with no parameters then all applications in memory are abandoned and the second number is lowered to equal the first.

<sup>58</sup> Chapter 4 - COMMAND and X.

<sup>59</sup> Chapter 8 - Drivers.

Next see an example of MEM /X from the same configuration as the previous one on the following page.

```

SpartaDOS X 4.46 | Tue 18-Jun-13 | 18:50:28
C:>MEM /X
Main: $1422,$1440,$2E8D
Ext: $708D,$7687,$708D
Use: BANKED, PORTB $2F
Top: $9C1F ($BC1F),$7FFF
Free: 34783 (42975),2424

16 banks total (256 KB)
4 banks free (64 KB)
C:>■

```

Fig. 33: Detailed Memory Information

The third number in 'Main' and 'Ext' indicates the top of the memory taken by program overlays (program modules loaded by applications) and is usually 0. 'Use' indicates after the use mode the type of extended memory. The example shows 'PortB' relating to XL/XE machines. 'AXLON' instead appears on machines using Axlon type extended memory, mainly 400/800. Next the page of extended memory is noted, which is used by SDX. 'Top' shows the highest available address in the main memory (before and after X.COM) and in the extended memory. 'Free' shows respective number of free bytes. Additionally, the total amount of the extended RAM in banks and kilobytes is displayed, besides the free amount. If you happen to use 65C816 high RAM, this amount will also be shown.

**Notes:** If a permanent driver is installed after LOADING an application held in memory, both low memory values will be raised above it and any application held in memory will become permanent.

Although there are two possible extended memory regions, SDX may use only one at a time. This is determined at boot-up time and depends upon the CONFIG.SYS file and/or the computer you are using.

Note that although the MEM command does not explicitly say what extended memory range is in use, it can be inferred by the addresses listed in the 'Ext:' field.

The ranges are as follows

- \$4000-\$7FFF Banked RAM (130XE or extended RAM computer)
- \$E400-\$FFBF OS RAM (not available on the Atari 800 computer)

The 'banks available' field indicates how many banks of RAM are still available for a ramdisk driver and/or BASIC XE extended mode.

**Note that you must have at least 4 banks available for BASIC XE extended mode.** Failure to pay attention to this fact may cause your system to crash (generally at the very worst time).

SDX currently supports up to 1024 KiB of extended memory with Port B types and 4032 KiB with Axlon types. Additionally, it indicates 65C816 high memory. The MEM



command should work with all known memory extensions and display their status properly.

This screenshot shows a system with 1 MiB of Port B type extended memory and 4 MiB 65C816 high memory, having this CPU type.

```
A:\MEM /X
Main: $1171,$1171,$0000
Ext: $749B,$749B,$0000
Use: BANKED, PORTB $0D
Top: $9C1F ($BC1F),$7FFF
Free: 35502 (43694),2916

64 banks total (1024 KB)
0 banks free (0 KB)

4096 KB linear RAM (64 segments)
A:\■
```

Fig. 34: Additional Memory Information On 65C816 High Ram



They represent the current directory shown in the directory window under the directory selector (→ Dir Commands).

- C **Copy** - copy the file under the file selector. It prompts for a destination drive, then a destination path. If copying to the same drive prompts will appear to insert the destination disk, and then to insert the source disk.
- ^C **^Copy** - copy all tagged files. Prompts are like with Copy.
- D **Delete** - delete the file under the file selector, but asks first for confirmation.
- ^D **^Delete** - delete all files tagged with tag character '◆'. Prompts to decide if all tagged files should be deleted with or without confirmation.
- E **Exec** - executes the file pointed to by the cursor.  
  
Exec is fully usable only when MENU.COM is started as a command processor (e. g. by adding SET COMSPEC=CAR:MENU.COM to your CONFIG.SYS).  
  
When MENU.COM was started as a program, the option can be used too, but with some limitations (e. g. it would not execute programs requiring the X command, nor would it allow the use of any command processor extensions such as RUNEXT or COMEXE).
- F **Filespec** - enter a filespec with wildcards to narrow down the logged (and displayed) files. Only legal filename characters and wildcards are allowed. Do not enter drive number or path here; instead use Log for that.
- L **Log** - change the logged drive number and/or path.
- P **Print** - print the file currently under the file selector. Useful only for ASCII text files unless a printer driver is installed which will print ATASCII graphics characters.
- ^P **^Print** - print the files currently tagged. A form feed is sent in between files.
- R **Rename** - rename the file under the file selector. As a reference, rename prompts with the present drive number, path, and filename. The new filename can then be entered directly under the old.
- S **Shell Cmd** allows to execute command processor commands.
- T **Tag** - tag (mark) the file under the file selector then move the file selector down one filename. A small tag character '◆' will appear to the right of the filename showing it as tagged.
- ^T **^Tag** - tag all files currently logged (in the current directory).
- U **Untag** - untag the file under the file selector. The tag character disappears and the file selector moves down to the next file.
- ^U **^Untag** - untag all files currently logged (in the current directory).

V **View** - display the content of the file under the file selector.

### Dir Commands

The directory selector indicates the current directory. While in the directory command menu, use the '↑↓' keys to move the directory selector up or down one directory at a time. When finished with the 'Dir Cmnds', press <RETURN> to go back to 'File Cmnds' or <ESC> for 'Xtra Cmnds'.

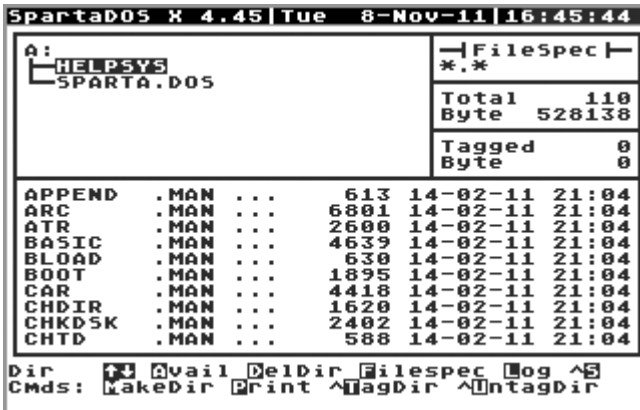


Fig. 36: Menu Program - Directory Commands

- A **Avail** - provides the amount of free space available on a drive. Enter a drive number and the free space will be shown in bytes.
- D **Del Dir** - use '↑↓' to move the directory selector. If the directory selected is empty (as shown in the file window), it can be deleted.
- F **Filespec** - enter a filespec with wildcards to narrow down the logged (and displayed) files. Only legal filename characters and wildcards are allowed. Do not enter drive number or path here; use Log instead. Go to the 'File Cmnds' to do any file operations other than tagging or untagging full directories.
- L **Log** - change the logged drive number and/or path.
- M **Make Dir** - creates a new subdirectory in the current directory selected. After the new directory is created, the system will re-log and then go back at the root of what was previously logged (always indicated by '>').
- P **Print** - prompts with two choices: Directory or Tree. Directory prints the list of the files as displayed in the file window. (If the display is set to show the short form files, they will be printed in one long list and not side by side as displayed.) Tree prints the directory map (tree structure) as displayed in the directory window.
- ^S **Shell Cmd** allows to execute command processor commands.
- T **Tag Dir** - tag all files in the current directory (under the directory selector).



### 4.39 MKDIR - make directory

---

#### Purpose

Create a subdirectory.

#### Syntax

MKDIR [d:]path

#### Alias

MD & CREDIR

#### Type

Internal

#### Related

CHDIR, RMDIR, PATH, RENDIR, DELTREE

#### Remarks

If you do not specify a drive, the default drive is assumed. This function is not supported by the ATARIDOS.SYS<sup>60</sup> driver even though subdirectories are supported by that driver.

Directories (also called subdirectories or folders) are used like file folders to organize your files. They also keep a large storage area fast. In a file cabinet it is much quicker to go to a file folder and search through a few documents, than a pile of all your documents. Computers work the same way. It is much quicker for DOS to go directly to a subdirectory and search through a few files than it is to search through one long file list.

Directory names are stored like filenames but marked with the +S attribute bit. They may not be renamed or deleted in the same way that files are. Please see the related commands.

```
MD TEST
MKDIR 3:>MODEM>TEST
CREDIR A:\SPARTA.SYS
```

The first example creates a subdirectory on the default drive called 'TEST'. The second example creates a subdirectory on D3: by the name of 'TEST' in the subdirectory called MODEM, which must already be there in the MAIN directory. The third example creates the directory named 'SPARTA.SYS' in the main directory of your boot drive A: (or D1: for that matter), which is needed to engage the config selector<sup>61</sup>.

---

<sup>60</sup> Chapter 8 - File System Drivers.

<sup>61</sup> Chapter 8 - Config Selector.

## 4.40 MORE - text viewer

---

### Purpose

Display the contents of the given text file.

### Syntax

```
MORE [<<][path]fname[.ext]
```

### Type

Internal

### Related

LESS, TYPE

### Availability

As of SDX 4.42.

### Remarks

The MORE command does exactly the same thing as TYPE fname /P (see TYPE). The MORE command is used most often as the final receiver of a data stream being sent through a pipe<sup>62</sup>, for example:

```
D1:ARC | MORE
```

is an equivalent to:

```
D1:ARC >>TESTIT
D1:MORE <<TESTIT
D1:DEL TESTIT
```

Such class of commands, which receive data from the standard input, process it, and dump the result to the standard output, is called 'filter commands'. Other filter commands are: LESS, FMT or INVERSE<sup>63</sup>.

**Note:** Invoked without any parameter the system will wait for an input from the CON: device. Pressing <CTRL><3>, <BREAK> or <RESET> aborts.

---

<sup>62</sup> Chapter 5 - Pipes & I/O Redirection.

<sup>63</sup> Chapter 9 - SDXTK

## 4.41 PATH - set search directory

---

### Purpose

Causes specified directories to be searched for commands before searching the current directory.

### Syntax

PATH [path\_string]

### Type

Internal

### Related

CAR, CHDIR, MKDIR, PROMPT, RMDIR

### Remarks

A list of drives and path names separated by semicolons may be specified. After this, when a command is entered, SD searches the named directories in the specified sequence (from path string) before searching the current directory of the drive that was specified (or implied). The current directory is not changed after the search.

Entering PATH with no parameters causes SDX to display the current setting of the PATH string.

It is recommended that you include 'CAR:' as a device in the search path as this device contains many external commands that you may need (such as X, CAR, MENU, DUMP, CHTD, etc.). It is also good practice to use '>' or '\ ' at the start of a device path to force a start at the MAIN directory. The command:

```
PATH A:>;1:>DOS;CAR:
```

sets the search to the root directory of drive A (alias drive D1:), the 'DOS' directory of drive 1, and the CARtridge directory.

The PATH command is really just a convenient form of the SET command, for example the above command could also be performed by:

```
SET PATH=A:>;1:>DOS;CAR:
```

The only way to clear the search path to search just the current directory (i. e. no search path at all) is the command:

```
SET PATH
```

When during boot up no path has been specified<sup>64</sup>, the system defaults to:

```
PATH CAR:
```

This means search the CAR: device first, then search the current directory.

---

<sup>64</sup> Chapter 5 - Default Batch File.



The current directory will always be searched last unless it is included in the path string. E. g.

```
PATH ;CAR: or PATH :;CAR:
```

The previous examples both mean the same thing; search current directory first, then CAR:, then current directory again. Remember that current directory is always searched last even if it was already searched. The stand alone ':' or a space, indicate the current directory<sup>65</sup>.

**Note:** While not required, it is strongly recommended that CAR: always be the first entry in the path string. The programs in this directory are called often. If any other devices are listed first, they will always be checked before the CAR: device, slowing system response considerably.

#### 4.42 PAUSE - a pause

---

##### **Purpose**

Suspends system processing and displays a prompt message.

##### **Syntax**

PAUSE [n]

##### **Type**

Internal

##### **Related**

DIR, TYPE

##### **Availability**

As of SDX 4.46 timing option.

##### **Remarks**

You can insert PAUSE commands within a batch file to provide the opportunity to change disks between commands or to step through a process, giving you time to read instructions, etc.

To resume execution of the batch file, press <RETURN>.

PAUSE now optionally accepts a number of seconds to wait, ranged from 0 to 65535.

**Note:** It is very dangerous to change disks during a PAUSE on the drive from which the batch file was running, or at least any changeable medium for that matter. If using PAUSE to change disks, run the batch file from a ramdisk or another drive that will not be changed.

---

<sup>65</sup> Chapter 5 - Search Path.

## 4.43 PEEK - display memory location content

---

### Purpose

Examine a memory location, perform a HEX conversion or symbol evaluation.

### Syntax

PEEK symbol | [[ $\$$ ]index:][ $\$$ ]address<sup>66</sup>

### Type

External - on CAR: device.

### Related

POKE, DPOKE (→ SDXTK)

### Availability

As of SDX 4.47 can access extended memory and 65C816 high RAM.

### Remarks

PEEK allows the examination of a memory location from the command processor. It is also useful as a quick DEC to HEX or HEX to DEC converter. (DEC means decimal or base 10; HEX means hexadecimal or base 16.) PEEK returns the dec and hex value of the location entered, the contents of location+1 in both dec and hex, the dec and hex value of the memory word stored in the location and location+1, and the ATASCII character representing the value of the location.

Available indices are   00 = main memory,  
                               02 = system extended memory,  
                               04 - max. 67 (255) = extended memory Port B (Axlon).

To peek 65C816 high ram type the respective number of digits, e. g. \$FF0000.

Addresses are assumed to be decimal unless preceded by a '\$', which indicates hex. Hex and dec values may be mixed.

The PEEK symbol command displays the address and memory index of the given symbol. The symbol will first be translated into an address, and then PEEK will proceed with this address as usual. Data to be displayed are fetched from the memory location the symbol points to.

**Note:** The commands Peek when invoked without any parameter will do nothing. Reason is to relieve system workload<sup>67</sup>; even no error message is displayed.

---

<sup>66</sup> See Chapter 6.4 for new functionality in calculating an address.

<sup>67</sup> Chapter 4 - Command.

#### 4.44 POKE - change memory location content

---

##### Purpose

Change the content of a memory location.

##### Syntax

POKE symbol | [[ $\$$ ]index:][ $\$$ ]address<sup>68</sup> [ $\$$ ]value

##### Type

External - on CAR: device.

##### Related

DPOKE (→ SDXTK), PEEK

##### Availability

As of SDX 4.47 can access extended memory and 65C816 high RAM.

##### Remarks

POKE allows you to change memory locations from the command processor. This can be useful in batch files and other applications. It is very easy to crash the system with this command if you do not understand what you are doing.

Available indices are   00 = main memory,  
                               02 = system extended memory,  
                               04 - max. 67 (255) = extended memory Port B (Axlon).

To poke 65C816 high ram type the respective number of digits, e. g. \$FF0000.

Addresses are assumed to be decimal unless preceded by a '\$', which indicates hex. Hex and dec values may be mixed.

A few examples of locations and useful values:

POKE 65 (soundr) 0=SIO sound off, 1=SIO sound on  
 POKE 77 (attract) 0=restart the attract timer at zero  
 POKE 82 (lmargin) n=number from 0 to 39 for left margin  
 POKE 83 (rmargin) n=number from 0 to 39 for right margin  
 POKE 559 (sdmctl) 0=screen off, 34=screen back on  
 POKE 710 (color2) 0=black, 53=red, 148=blue  
 POKE 730 (keyrep) 1=hyper, 3=fast, 5=normal (XL/XE only)  
 POKE 731 (noclik) 0=normal, 1=speaker off (XL/XE only)  
 POKE 752 (crsinh) 1=cursor off, 0=cursor on  
 POKE 702 (shflok) 0=lower case, 64=upper case

**Notes:** Always PEEK a location before POKEing a value into it. Recovering by POKEing the old value back in is possible, unless the computer has crashed.

The command Poke when invoked without any parameter will do nothing. Reason is to relieve system workload<sup>69</sup>, even no error message is displayed.

---

<sup>68</sup> See Chapter 6.4 for new functionality in calculating an address.

<sup>69</sup> Chapter 4 - Command.

## 4.45 PROMPT - set system prompt

---

### Purpose

Change the system prompt.

### Syntax

PROMPT [prompt\_string]

### Type

Internal

### Related

PATH

### Remarks

The text in prompt string is taken by SDX to be the new system prompt. Special meta-strings can be embedded in the text in the form '\$c' where 'c' is one of the following characters:

- L display current drive letter ('A' through 'O') and a following colon (e. g. 'C:')
- N display current drive number ('1' through '9') and a following colon (e. g. '3:')
- P display path on current drive
- D display current date
- T display current time
- R display an EOL character (advance to next line)

If no parameter is specified, the current prompt string will be displayed.

For example the command

```
PROMPT $L$P\
```

will display a prompt in the form 'B:\DOS\' assuming the current drive is 2 and the current path is 'DOS'. Also, the '\_' character will display as a space rather an underline. Thus a prompt can end in a space.

The PROMPT command is just a convenient form of the SET command.

```
SET PROMPT=$L$P\
```

would perform the same. The default value of the 'prompt' variable is 'D\$N', which displays the same prompt like previous versions of SDX. If the \$PROMPT variable is not defined, SDX will prompt with a '>' character. The only way to clear the \$PROMPT variable is with the command

```
SET PROMPT
```

Because the command processor automatically converts all lower case characters to upper case prior to processing, normal lower case cannot be used in the prompt. Inverse and inverse lower case characters and cursor control keys (preceded by the escape key) may be used in the text part of the prompt.

**Notes:** Depending on the used path string it may no longer be possible to re-enable the content of formerly used command lines by moving the cursor and pressing just <RETURN>. DOSKEY<sup>70</sup> comes in handy then.

When using the '\$P' meta-string in the prompt, the default drive will be read each time the prompt is printed. This will cause an error to be printed within the prompt if there is no disk or a bad disk in the default drive, or if the disk is of a filesystem not recognized by SDX. To use AtariDOS format disks with SDX you must install the ATARIDOS.SYS driver.

Using the '\$P' meta-string in the prompt can also cause problems when attempting to park a hard drive, since the drive will be 'unparked' to read the path when the prompt is printed. The solution to this is to set the environment variable \$PROMPT to a value not containing '\$P'. Since drives are usually parked only when you are through using the computer, you may simply clear the \$PROMPT before parking the drive. You could set up a batch file to clear the prompt variable and then park the drive to simplify this operation.

```

SpartaDOS X 4.45 | Tue 8-Nov-11 | 16:30:18
D1: PROMPT
D$N:

D1: SET PROMPT=$D_DRIVE_$L$R

 8-11-11 DRIVE A:
PROMPT
$D_DRIVE_$L$R

 8-11-11 DRIVE A:
SET PROMPT=$L$P\

A:\HELPSYS\PROMPT Drive_$L$P\
DRIVE A:\HELPSYS\CD ..
DRIVE A:\PROMPT $L$P>
A:>■

```

Fig. 38: Customize Your System Prompt

<sup>70</sup> Chapter 8 – See under Other Drivers.

## 4.46 PWD - print working directories

---

### Purpose

Output a list of current working directories.

### Syntax

PWD

### Type

External - on CAR: device.

### Related

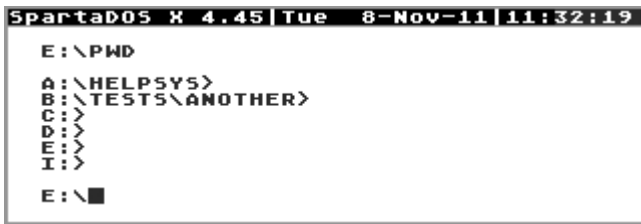
DIR

### Availability

As of SDX 4.43.

### Remarks

A quick overview of the current working directories on all valid drives.



```
SpartaDOS X 4.45 | Tue 8-Nov-11 | 11:32:19
E:\PWD
A:\HELPSYS>
B:\TESTS\ANOTHER>
C:\>
D:\>
E:\>
I:\>
E:\█
```

Fig. 39: Display The Current Working Directories

#### 4.47 RENAME - rename files

---

**Purpose**

Change the name of one or more files.

**Syntax**

```
RENAME [d:][path]fname[.ext] fname[.ext]
```

**Alias**

REN

**Type**

Internal

**Related**

MENU

**Remarks**

Wildcards may be used in both filespecs. A device and path may only be specified on the first filename (the old name filespec). Filenames must be specified for both source and destination names, otherwise an error will occur. The rules for the use of wildcards are the same as for the COPY command. Here are a few examples:

```
RENAME *.BAK *.DOC
```

The above command changes all the extensions to 'DOC' of files that previously had extensions of 'BAK'.

```
RENAME AC*.* *.XX
```

This command changes the extension of all files beginning with 'AC' to 'XX'.

**Note:** Now with SDX 4.4x there is a check for already existing filenames. Error 151 'File exists' will be displayed if an attempt is made to change a filename to one that already exists and no other action is performed. Please use another name then.

## 4.48 RENDIR - rename directory

---

### Purpose

Change the name of a directory.

### Syntax

RENDIR [d:][path]dir\_name\_old dir\_name\_new

### Type

Internal

### Related

MKDIR, RMDIR

### Availability

As of SDX 4.40.

As of SDX 4.42 as internal command.

### Remarks

The command does to directories, what RENAME does to files. The same rules apply.

```

D0:DIR
Volume:      X Disk 0
Directory:   MAIN
SD_DOC      <DIR>      6-05-12  14:08
SD_MAN      <DIR>      6-05-12  14:09
SD_TK       <DIR>      6-05-12  14:07
          391 FREE SECTORS
D0:RENDIR SD_TK SDX_TK
D0:DIR
Volume:      X Disk 0
Directory:   MAIN
SD_DOC      <DIR>      6-05-12  14:08
SD_MAN      <DIR>      6-05-12  14:09
SDX_TK      <DIR>      6-05-12  14:10
          391 FREE SECTORS
D0:█

```

Fig. 40: RENDIR Command



## 4.49 RMDIR - remove directory

---

**Purpose**

Delete an empty subdirectory from the specified drive.

**Syntax**

RMDIR [d:]path

**Alias**

RD & DELDIR

**Type**

Internal

**Related**

CHDIR, DELTREE, MKDIR, PATH

**Remarks**

Only empty directories can be removed. The last directory name in the path is the directory to be removed. This function is not supported by the ATARIDOS.SYS driver.

```
RD TEST
DELDIR 3:>MODEM>TEST
```

The first example removes the subdirectory called 'TEST' on the default drive from the current directory. Error 167 will occur, if the directory has files in it. The second example removes a subdirectory on drive D3: by the name of 'TEST' in the subdirectory called MODEM which is in the MAIN directory.

**Note:** If a file has been opened for write or update but not properly closed (usually by hitting reset or losing power while it is opened) its entry in the directory will not be removed, although it may not show in a listing. A subdirectory containing a 'phantom' entry of this type can not be deleted. 'CleanUp X' cures such problems automatically.<sup>71</sup>

## 4.50 RS232 - load RS232 driver

---

### **Purpose**

Load the RS232 handler from a P:R: Connection or the Atari 850 interface.

### **Syntax**

RS232

### **Type**

External - on CAR: device.

### **Remarks**

You need to use this command prior to using a P:R: Connection or Atari 850 interface unless the program you are going to use does this automatically. Try your program without RS232 first. You should hear a beep on your monitor (TV) speaker if the handler loads. If not, an error will occur. Type this command and run your program again.

Avoid loading the RS232 handler more than once. Your system may crash if you load several copies of the RS232 handler into memory, since MEMLO is raised each time.

**Note:** Booting the driver from the Atari 850 more than once is not possible since it can only be loaded once after power has been switched on.

## 4.51 SAVE - save binary data

---

### Purpose

Save binary data from memory to disk.

### Syntax

SAVE [d:][path]fname[.ext] [\$]address [\$]address

### Type

External - on CAR: device.

### Related

LOAD, BLOAD

### Availability

As of SDX 4.47 the SAVE command is now external on CAR: device.

### Remarks

Useful tool when used in conjunction with the LOAD command for de-segmenting MAC/65 files or to save a snapshot of memory for debugging purposes.

```

SpartaDOS X 4.45 | Tue 8-Nov-11 | 11:18:24
A:\SAVE TEST.DAT $4000 $4007
A:\MDUMP $4000 8
 4000- AD F8 0C 85 15 AD F9 0C
A:\DUMP TEST.DAT
0000- FF FF 00 40 07 40 AD F8
0008- 0C 85 15 AD F9 0C
A:\

```

Fig. 41: SAVE Command

Addresses are assumed to be decimal unless preceded by a '\$', which indicates hex.

## 4.52 SET - set environment variable

---

### Purpose

Display the values of all environment variables, and optionally set an environment variable to a specified value.

### Syntax

```
SET [var[=env_string]]
```

### Type

Internal

### Remarks

Environment variables are global strings that can be used by a program to communicate to another. E. g. the \$CAR variable tells the CAR command where the memory-save file is. These three forms are available:

**SET** displays the content of all environment variables.

**SET CAR=A:CAR.SAV** sets the variable \$CAR to the value 'A:CAR.SAV'.

**SET CAR** deletes the environment variable \$CAR from the system. (This will cause the CAR command to not use a memory-save file.)

SET may be used in CONFIG.SYS, AUTOEXEC.BAT or via command line.

```

SpartaDOS X 4.45 | Tue 8-Nov-11 | 11:36:14
A:\SET
BOOT=A:>
SYSERR=CAR:SYSERR.MSG
MAXDRU=0:
RAMDISK=I:>
BASIC=I:>BASIC.SAV
CAR=I:>CAR.SAV
TEMP=I:>
SCRDEF=14,6
MANPATH=A:\HELPSYS
PAGER=LESS;/C;%%
PATH=CAR:;D2;;D9:
PROMPT=$L$P\

A:\█
  
```

Fig. 42: Display The System Variables

The '\$'-character can be used to terminate a name of an environment variable, so that it can be referenced not only as \$var but also as \$var\$. This makes it possible to insert a variable into a longer string, which does not contain usual path separators.

### 4.53 SETPATHS - set current path

---

**Purpose**

Set current directories on the specified drives.

**Syntax**

SETPATHS [d:][path] | fname[.ext]

**Type**

External - on CAR: device.

**Related**

CHDIR

**Availability**

As of SDX 4.40.

**Remarks**

After DOS startup, the current directory on every disk points to the main directory. To change them automatically to required subdirectories, the SETPATHS command should be invoked from the AUTOEXEC.BAT file. As a parameter the name of a text file should be given to SETPATHS. This file should contain valid subdirectory specifications in consecutive lines. For example:

```
A:>DOS>  
B:\UTILS\  
C:>PRG>SRC>
```

If the paths specified this way do exist, the respective directories will become the current ones on the given drives upon completion of the AUTOEXEC.BAT file.

Alternatively the required path can be specified directly as a command line argument.

## 4.54 SIOSET - SIO speed control

---

### Purpose

SIO.SYS serial speed control.

### Syntax

```
SIOSET [d:] [type [usindex]]
SIOSET REFRESH [0|128]
SIOSET NMI [index]
SIOSET WAITACK [[[$]index]
```

### Type

External - on CAR: device.

### Availability

As of SDX 4.40.

### Remarks

SIOSET manages an advanced control of the serial protocol for the SIO.SYS driver<sup>72</sup>. Typically, the serial transmission parameters are determined automatically on the first access of a drive. Later on there is no need to change them. Sometimes however, e. g. when a drive was changed to another type at run time, you may want to change the parameters manually.

SIOSET invoked with no arguments displays the current configuration for all SIO drives accessed so far, otherwise a hyphen is noted. 'SIOSET 4' e. g. shows the information just for drive #4. The DF command accesses all drives, which helps to see their current configuration with SIOSET.

The following features may be set using the type parameter:

RESET	The transmission parameters for the given drive # are cleared; they will be determined on the next I/O request sent to that drive.
NORMAL	The drive now works at SIO standard baud rate.
XF	The drive uses the XF551 protocol.
INDUS	The drive uses the Indus protocol (LDW Super 2000, CA 2001).
US	The drive uses the Ultra Speed protocol.

The manually issued settings will be confirmed by displaying them.

Note that e. g. 'SIOSET 3 US' will set drive #3 to Ultra Speed, determine its speed ability and enable it by automatically setting the appropriate us index.

With Ultra Speed the additional parameter 'usindex' allows to determine the serial speed manually. Make sure to set a legal value for the addressed drive.

The SIO driver before SDX 4.47 periodically re-queried drives for high speed by default. This has been disabled since it caused problems with some drive types. Invoke SIOSET REFRESH to see the new default setting (0=no re-query).

Use SIOSET REFRESH 128 to enable the former behavior. However, the refresh may affect manual high speed settings, since it checks the drive on the next access and resets the us index to the detected value.

The option NMI allows the user to check and set the lowest high speed index at which NMI interrupts are kept intact. This index defaults to 8 (3x SIO). Check your NMI setting by typing SIOSET NMI.

Below the set value NMIs are turned off to keep time requirements for high speeds. Increase the threshold if you experience transfer errors or lock-ups at the corresponding speed. Decrease the set value if you want to keep NMIs turned on, as is required by some programs. Be advised that this may affect transfer reliability.

The option WAITACK allows the user to check and adjust the response timeout for SIO devices that do not conform fully to SIO standards. Therefore the use of modified operating system routines for such devices is now obsolete. Invoking the option without an index given will show the current setting.

2           ATARI operating system default value.

16           Setting needed for the SIO2BT interface connection.

**CAUTION:** SIOSET does not work, if SIO.SYS was loaded using the '/A' or '/C' option!

**Notes:** The SDX high speed SIO driver currently handles SIO transfers stable as low as ultra speed index 3 on stock A8 machines. The then available SIO speed in your system depends on your hardware setup.

Sometimes drives in AspeQt/RespeQt do not respond and SDX reports error 138. SIOSET comes in handy then. Check the speed setting in AspeQt/RespeQt and issue the related parameters, e. g.:

```
SIOSET 8 US $03
```

This is not a bug in SDX but seems to be a problem of the host system running.

When using the Ultra Speed driver from the KMK/JŽ IDE V. 2.0 Plus, the ultra speed index will be handled automatically by its driver. SIOSET will neither affect this driver nor will it show the settings. This driver can keep NMI interrupts on even when running at US index 0.

## 4.55 SORTDIR - sort filenames

---

### Purpose

Sort filenames in directories by name, type, date or size.

### Syntax

```
SORTDIR [d:][path] [/N|T|S|D|X]
```

### Type

External - on CAR: device.

### Availability

As of SDX 4.41 originating from the SDK.

As of SDX 4.43 written completely new from scratch to overcome some disadvantages.

As of SDX 4.47 MyDOS formatted media (except 90/180 KiB) will be handled correctly.

### Remarks

The command reads the specified directory, sorts it using the specified criteria, and then writes it back. The criteria can be:

- /N - sort by name
- /T - sort by type
- /S - sort by size
- /D - sort by date and time
- /X - sort in descending order

When the path specification is omitted, the current directory is sorted then. At least one criterion is obligatory. SORTDIR invoked without arguments displays a brief copyright information and lists available options.

When the files are sorted by name, the file type is a second priority. When sorting by type, the second priority is the file name. When sorting by size, the second priority is the name, and the type is the third. Digits are prior to letters. Everything is sorted in ascending order by default, the [/X] switch reverses that order.

SORTDIR refuses to sort AtariDOS 2 type formatted media, when detected.

**Note:** Though sorting directories on other file systems may look successful, it is not recommended. Copy files to be sorted to a SDFS medium first to avoid problems.



## 4.56 SWAP - swap drives

### Purpose

Swap (re-map) your drive configuration.

### Syntax

SWAP [d d]

### Type

Internal

### Remarks

SWAP, without any parameters, will display the drive map list showing drives 1 through 15. The default is A: = 1, B: = 2, etc.

To swap drives D2: and O:, type

```

SpartaDOS X 4.47 | Tue 3-Dec-13 | 13:25:43
A:\SWAP 2 0
A:\SWAP
A: = drive 1
B: = drive 15
C: = drive 3
D: = drive 4
E: = drive 5
F: = drive 6
G: = drive 7
H: = drive 8
I: = drive 9
J: = drive 10
K: = drive 11
L: = drive 12
M: = drive 13
N: = drive 14
O: = drive 2
A:\█

```

Fig. 43: SWAPPING Logical Drives

The order is not important, so 2,0 is equivalent to O,2. Please remember drive ids higher than 9 can only be referred to by letters.

The drives will stay mapped that way until remapped or a COLD start occurs. Note that you may use letters or numbers to reference a drive and that no colon (':') follows the drive specifier.

**Note:** SWAP works in addition to the MAP command and drive remapping set by MIO, Blackbox, MSC or alike<sup>73</sup>. So take that into consideration when using such hardware. It is very easy to lose track of which floppy disk drive, ramdisk, or hard drive partition is at what logical drive.

## 4.57 TD - time/date display

---

### Purpose

Turn on and off a time/date display line on top of your screen.

### Syntax

TD [ON|OFF]

### Type

External - on CAR: device.

### Related

CHTD, DATE, TIME, DAYTIME

### Remarks

This command is much like the KEY command in the way it links into your computer.

Without a proper clock driver<sup>74</sup> installed this command will produce meaningless results. By default, one of the drivers bundled with SDX will be installed during boot up, but can be overridden by creating a custom 'CONFIG.SYS' file to change the preset.

The format in which the time is displayed depends on the value of the \$DAYTIME variable<sup>75</sup>. Use the SET command to enable the preferred display format.

An additional feature is the 'X' letter in the time/date display line, which reflects the Caps/Inverse state of the keyboard.

TD may be installed without switching the time/date display line on by just typing TD without any parameter. TD ON may be incompatible with some programs. If you are having problems with a program, try TD OFF, or do not install it at all.

**Note:** Additionally, Z.SYS<sup>76</sup> enables to read and set time/date from within a program written e. g. in BASIC.

---

74 Chapter 8 - Time Keeping Drivers.

75 Chapter 8 - Important system variables.

76 Chapter 4 - Date.

## 4.58 TIME - show and set time

---

### Purpose

Display the current time and set the time.

### Syntax

TIME [/T][hh[:mm][:ss]]

### Type

External - on CAR: device.

### Related

CHTD, DATE, TD

### Remarks

The command TIME in SDX V 4.4x uses a 24 hour format only.

```

SpartaDOS X 4.46 | Sat 22-Jun-13 | 8:43:49
D1:TIME
Current time is: 20:22:19
Enter new (HH:MM:SS):

D1:TIME 8:42

D1:TIME /T
8:42:53

D1:DATE
Current date is: 25-01-06
Enter new (DD-MM-YY):

D1:DATE 22-6-13

D1:DATE /T
22-06-13

D1:█

```

Fig. 44: Using The TIME Command

Enter the new time or press <RETURN> to keep the current settings. The time format - 'HH' for hours, 'MM' for minutes, and 'SS' for seconds - is obligatory to change the settings. Type 'HH' to change the hours, 'HH-MM' to set hours and minutes, all to change the seconds too. The space key is a legal delimiter.

When fed with a valid time value in the command line, this value will be set as current.

The /T parameter causes to display the current time. No prompt will appear.

Without a proper clock driver<sup>77</sup> installed this command will produce meaningless results. By default, one of the drivers bundled with SDX will be installed during boot up, but can be overridden by creating a custom 'CONFIG.SYS' file to change the preset.

**Note:** Additionally, Z.SYS<sup>78</sup> enables to read and set time/date from within a program written e. g. in BASIC.

---

<sup>77</sup> Chapter 8 - Time Keeping Drivers.

<sup>78</sup> Chapter 8 - Time Keeping Drivers - Z.SYS.

## 4.59 TYPE - display file contents

---

### Purpose

Display the contents of a specified file.

### Syntax

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

### Type

Internal

### Related

COPY, DUMP, MENU, PAUSE, LESS, MORE

### Remarks

Displays any file and is not limited to a maximum line length as was the case with SD 3.2. Press <CTRL><1> to stop and start the display. Attributes as with the DIR<sup>79</sup> command apply - the default attributes are '-HS'.

If you include a '/P' parameter, the TYPE command will wait for a key press after each 23 lines of text.

```

MAN Command
=====
Purpose
-----
Starts the documentation viewer.

Syntax
-----
MAN [/?]
MAN [fname]/P]

Type
-----
External - on device CAR:

Availability
-----
As of SpartaDOS X 4.40.

Remarks
-----
MAN.COM is a basic text viewer, whose
Press <RETURN> to continue █

```

Fig. 45: TYPE Used To Display The File MAN.MAN.

## 4.60 UNERASE - restore erased files

### Purpose

Restore files previously erased (if possible).

### Syntax

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

### Type

External - on CAR: device.

### Related

ERASE

### Remarks

Wildcards are permitted. A prompt will appear for each recoverable file matching the entered filespec, if to restore it or not. If erased files are suspected to exist in that directory but have not been detected by UNERASE, they are not recoverable for one of two reasons. Either the file's directory entry has been allocated to another file which was copied to the directory after the original file was ERASEd, or a sector of the file has been allocated to another file since the original file was ERASEd.

```

D1:DIR
Volume:   TESTS
Directory: MAIN
AUTOEXEC  BAT      65  25-01-06  20:20
MAKEDAT   BAS     147  16-03-09  21:05
TESTS     <DIR>   25-01-06  20:16
1411 FREE SECTORS

D1:UNERASE *.*
Restore TEST.DAT? Y
Restore ABZDE.DAT? N
Restore ABCRAIG.DAT? N
Restore TEST.DZI? N
Restore ABCDE.ICD? N

D1:■

```

Fig. 46: Restore Files Using UNERASE

**Notes:** UNERASE.COM in SDX 4.2x contains a serious bug, that is fixed now. The test distributed with Nelson Nieves' NNTOOLS now passes without errors.

## 4.61 VER - SDX version information

---

### Purpose

Display the current version number and date of the cartridge.

### Syntax

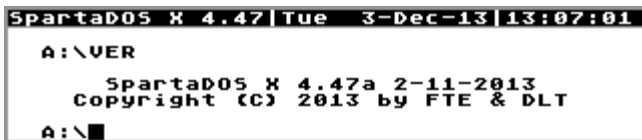
VER

### Type

Internal

### Remarks

This command will show the version number, revision date, and copyright notice as displayed when the cartridge is booted.



```
SpartaDOS X 4.47 | Tue 3-Dec-13 | 13:07:01
A:\VER
SpartaDOS X 4.47a 2-11-2013
Copyright (C) 2013 by FTE & DLT
A:\█
```

Fig. 47: Display The SDX Version Information

## 4.62 VERIFY - write verify

---

### Purpose

Turn write verify on or off.

### Syntax

VERIFY [ON|OFF]

### Type

Internal

### Remarks

When ON, SDX performs a verify operation following each disk write operation, to verify that the file system data just written can be read without error. Because of the extra time required to perform the verification, the system runs slower when programs write data to disk, especially on real floppy disk drives. This command is typically used when drive problems occur. The default is OFF.

VERIFY command displays the current status of the write verify feature whenever changed or when the command is invoked without parameters.

### 4.63 X - execute without cartridge

---

#### Purpose

Execute a program which requires that no cartridges are installed, e. g. Turbo-BASIC XL, most binary files, etc.) using specific loading modes.

#### Syntax

X [/C|L S|A|N|I][mode] [d:][path]fname[.ext] [parameters]

#### Type

External - on CAR: device.

#### Remarks

There are four possible environments for a program to be run:

1. with internal BASIC present (via BASIC command)
2. with a CARtridge present (via CAR command)
3. with the SDX library enabled (just type the program or command name)
4. with no cartridges present (via the X command)

The first three modes use the SDX library to perform various DOS functions, including loading and running the command. The fourth mode, however, cannot use the library without moving or disabling the screen! Thus, the following features are disabled when programs are run with this command:

- The mini-buffers are not used - single byte get/put will be very slow (this is extremely rare since most programs that use single get byte and put byte are in BASIC or use a cartridge)
- Since the library is disabled, only standard binary files are loadable - SD external commands such as FIND and MENU cannot be run.
- I/O redirection is severely hampered because it must use the library. In doing this the screen will flicker rapidly.

The general rule of thumb is: 'If a program does not work with a cartridge installed, prefix the command with the X command, otherwise, just type the command.'

The '/C' switch causes X.COM to clear the following memory locations before loading the program: \$80-\$FF, \$0400-\$06FF, and everything from MEMLO to MEMTOP. This can prove useful when running some programs.

As of SDX 4.47 this operation may be reverted now by using the \$X environment variable: 'SET X=C' will do.

As of SDX 4.47 the '/L' switch, enables X.COM to use different loading modes. For details please see the LOAD Command. The default loading mode is 0 (=SDX).

While the called program is running, X.COM will remain resident in memory, so MEMLO will be slightly higher until the program exits to the command processor.

X.COM now does not change ANTIC's DMA state when switching the cart bank on/off (this is VBXE 80-column display support).

X.COM will always setup the OS memtop (\$02E5) value as if the GR.0 console was active, and if appropriate, allow the E: driver to change it later. This prevents problems with various screen drivers using the memtop value.

Executing a cold start (a jump to \$E477) while using X.COM will disable the SDX cartridge and the external cartridge, if present.



## 5 The Command Processor - Advanced Features

The SDX command processor has been greatly enhanced with more sophisticated batch files, command line I/O redirection, user definable prompts, command search paths, and more. This chapter discusses these features and gives many examples. Most of these features are either new to SD or have been enhanced over prior versions.

### 5.1 Running Programs

If a program requires free memory in the area normally occupied by the I/O library (\$A000-\$BFFF), it should be executed with the X command. You can also precede the program name with the hash mark instead, for example, typing at the DOS prompt **#MYPROG** will have the same effect as **X MYPROG**.

### 5.2 Batch Files

Batch files are simply a list of SD commands that may be fed to the command processor from a text file. Parameters may be passed to the batch file by including them on the command line following the batch file name. The syntax is

```
-fname [parm1 parm2 ... parm9]
```

The filename ('fname') is assumed to have an extension of '.BAT' although this assumption may be overridden by including an extension on the command line. The parameters ('parm') are optional. A text line starting with a semicolon (;) is understood as a comment and skipped without parsing.

Batch files may be concatenated by calling another batch, but be aware that nesting is not allowed. Every command after the call of another batch file will be skipped. Therefore it is recommended to place the call of another batch as the last command.

The command **ECHO OFF** used in a batch file prevents displaying the commands read from the batch file on the screen. **ECHO ON** enables the echoing. The default is ECHO OFF. You have to write **ECHO ON** explicitly in the batch file, if you want to see what is being done.

The following is an example of a batch file named 'TEST.BAT' that accepts two files on input and creates an output file containing the content of both source files.

```
COPY %1 %3  
COPY %2 %3/A
```

Now, the command

```
-TEST FILE1 FILE2 OUTPUT
```

will concatenate the files 'FILE1' and 'FILE2' to the file 'OUTPUT'.

You may pass up to 9 parameters (numbered '%1' to '%9') to a batch file. The parameter '%0' is the name of the batch file (in the previous case, this would be 'TEST'). The '%' parameters may appear anywhere in the batch file and may be surrounded by or imbedded in text (i. e. no spaces need to precede the '%' character).

The batch file parameters are automatically saved in environment variables '\_x1' where 'x' is the parameter number. Due to the command processor's non-resident nature, the parameters need to be saved somewhere - environment variables are a perfect place. This also means that the number and size of parameters is limited to a total of 256 characters less overhead (the '\_x1=' string) and space used by other environment variables.

### 5.2.1 Default Batch File

On the first entry the command processor searches in the main directory of the boot drive for a batch file called 'AUTOEXEC.BAT'. Put any system setup commands needed into this file.

The variable \$BATCHE will be read by the command processor just before it prints its prompt. If this variable exists and contains a filename, this file will be executed as a batch file. As soon as the command processor reads the variable, it will delete it from the list of environment variables. This is how the system passes the 'AUTOEXEC' filename to the command processor when it is loaded for the first time. Upon booting, the variable \$BATCHE can be used to cause a batch file with a different name to be executed. Use the line:

```
SET BATCHE=d:filename.BAT
```

in CONFIG.SYS.

**Note:** 'BAT only' will be displayed when commands only legal in batch files were erroneously issued on the command line.

### 5.2.2 Conditionals

The external commands IF, ELSE, FI allow simple conditional expressions in batch files. The general syntax is:

```
IF [NOT] [expression] operator parameter
  ...
FI

or

IF [NOT] [expression] operator parameter
  ...
ELSE
  ...
FI
```

Such conditionals can be nested, up to 255 levels of nesting is allowed.

#### 5.2.2.1 IF EXISTS Conditional

The operator EXISTS allows to check the presence of the specified file or directory on a disk. For example, 'IF EXISTS FOO.BAR' is true, when a file named 'FOO.BAR' exists in the current directory. NOT negates the result, so IF NOT EXISTS FOO.BAR will be false in this case. It can only see regular files by default, to check if a directory exists, you have to specify the attribute the usual way: IF EXISTS +S FOO returns true, if a subdirectory

named 'FOO' exists in the current directory. The following example can be saved as a batch file named X.BAT somewhere in your \$PATH:

```
IF EXISTS %1.ARC
  IF NOT EXISTS +S %1
    ECHO Creating dir %1
    MD %1
    ARC X %1 %1>
  ELSE
    ECHO %1 already exists
  FI
ELSE
  EXIT 170
FI
```

Then, typing at the command prompt **-X FILES** allows to unpack the archive FILES.ARC into a subdirectory automatically created for this purpose.

#### 5.2.2.2 IF ERROR Conditional

The operator ERROR allows to check error conditions. 'IF ERROR' is true, if any error has been recorded by the system. Similarly, 'IF NOT ERROR' is true, when there was no error condition. Alternatively you can specify the exact error number: 'IF ERROR 170' returns true, if the error that occurred last was 'File not found'.

**Note:** The ERROR keyword uses a system variable called ERRNO, that is only written to by the system library on any error condition and never cleared. This means, that the error code the variable contains may not necessarily have been generated by the command that was executed last. Therefore it is good practice to do 'SETERRNO 0' before running a command that is about to be checked later with IF ERROR.

#### 5.2.2.3 IF INKEY Conditional

When this operator is encountered, the batch file execution stops and the system waits for a key to be pressed. The expression 'IF INKEY' is true, when the user hits any key (except Break - it is false then). You may also specify the key to be waited for. To accomplish that, an argument must be specified to the 'INKEY' keyword, either a numeric ATASCII code of the key, or its text value in single quotes. For example, when you want the batch file to wait for the 'A' key, the command doing that may be in any of the forms below:

```
IF INKEY 65
IF INKEY $41
IF INKEY 'A'
```

This serves well in simple comparisons, but for more complex tasks, like a menu with many options, you should use the INKEY command described below.

**Note:** The 'INKEY' operator is part of the IF command and should not be confused with the 'INKEY' command.

### 5.2.2.4 IF FILE conditional

In a batch file this conditional type enables to compare files in terms on time stamp and length:

```
IF FILE fname1.ext op fname2.ext
```

The operator 'op' may be one of the following:

- NT (newer than),
- OT (older than),
- EA (equal age),
- LT (longer than),
- ST (shorter than),
- ES (equal size).

```
Example: IF FILE FOO.BAR OT BAR.FOO
          ECHO FOO.BAR OLDER THAN BAR.FOO
        ELSE
          ECHO FOO.BAR NOT OLDER THAN BAR.FOO
        FI
```

When a specified file does not exist, the condition is not met.

### 5.2.3 Existence of environment variables

A method to check for an environment variable is:

```
IF [NOT] [name of environment variable]
    ...
FI
```

The expression will be true if the variable exists, or false otherwise.

### 5.2.4 Comparisons

Checking the value is accomplished with the double equation sign (==), employed here as a comparator. For example, IF DAYTIME=="2" is true, if the environment variable \$DAYTIME contains the text '2'. Analogically, IF "%0"=="TEST" is true, if the batch file that contains the conditional, is named 'TEST.BAT'. There is no separate 'not equal' operator, this condition can be checked by combining the double equation sign and the logical negator NOT. So, IF NOT DAYTIME=="2" means 'if \$DAYTIME is not equal to 2'.

### 5.2.5 Loops

The batch commands FOR and NEXT allow to make counted loops in batch files. The syntax is:

```
1) Numeric: FOR var=start_value TO end_value
             [bat commands]
             NEXT var
```

This one allows to repeat a set of commands the required number of times.

```
Example: FOR NUM=0 TO 15
         TYPE NUL: >>FOO$NUM$.BAR
         NEXT NUM
```

It will generate 16 zero-length files named FOO0.BAR, FOO1.BAR, FOO2.BAR, ... and so on up to FOO15.BAR.

```
2) String: FOR var IN [+|-][AHPS] [d:][path]filemask [/NX]
           [bat commands]
           NEXT var [+|-][AHPS] [[d:][path]filemask] [/NX]
```

This one allows to apply the required commands to files located in 'd:path' and selected by the mask 'filemask' and attributes as shown above. Example:

```
FOR FILE IN + CAR:
  TYPE + CAR:$FILE >>$FILE
NEXT FILE + CAR:
```

This allows to batch-copy all files located on CAR: to current directory on current drive. The switches operate as follows:

/N - only the file name (without extension) will be assigned to var  
 /X - only the extension will be assigned to var  
 By default, the 'filename.ext' is stored in var.

The additional parameters for the NEXT keyword, after the variable name, may be omitted, if the corresponding FOR-line specifies no attributes and no switches, and the filemask specified there is '\*.\*'. Otherwise the NEXT-like should repeat the attribute, disk, filemask etc. specified for the corresponding FOR-line

The maximum size of the 'var' name is 8 characters. It is good idea to begin with % any variable names to be used locally within a BAT file, as such ones are automatically deleted after the batch file exits. For the examples above, use %NUM and %FILE instead of NUM and FILE.

### 5.2.6 GOTO Jumps

The 'GOTO' command allows to make a jump within the batch file. The syntax is:

```
GOTO label
```

This simply transfers the batch file execution to the line following the one, that contains the 'label' at its beginning. An example definition of a label may look as follows:

```
:LABEL
```

To use the label's name as a parameter to the 'GOTO' keyword just omit the colon.

**Note:** Bear in mind, that a GOTO searching for its label always goes through the entire batch file from the first line to the last. This means, that the farther the label is in the batch file, the slower a 'GOTO' jump will be.

### 5.2.7 INKEY Command

This command stops the BAT file execution, waits for a key to be pressed, and, when pressed, assigns the corresponding letter to the specified environment variable. This value can be read within a comparison in the 'IF' statement, This way it is possible to create a menu with more options:

```
:MENU
CLS
ECHO A. Option no. 1
ECHO B. Option no. 2
ECHO C. End
INKEY %KEY
  IF %KEY=="A"
    ECHO Option 1 was selected
  FI
  IF %KEY=="B"
    ECHO Option 2 was selected
  FI
  IF %KEY=="C"
    SET %KEY
  EXIT
FI
PAUSE
GOTO MENU
```

**CAUTION:** Data is written to the environment storage and this space is only 256 bytes. To avoid filling it with unnecessary garbage in SDX versions before 4.42, it is good practice to delete all variables created in your batch file before exiting it; just as shown in the previous example (the fifth line counting from the end).

**Note:** The 'INKEY' command should not be confused with the 'INKEY' operator, which is part of the 'IF' command.

As of SDX 4.42 an extension 'ENV.SYS' is available, which provides as much as 15 KiB for environment variables, and the Command Processor, upon BAT file completion, automatically removes variables starting with '%', ':', and '\_'. The latter two types are created implicitly by the Command Processor and batch command binaries. The third type, the one starting with '%'-sign, is at the user's disposal.

### 5.2.8 Procedures

It is now possible to define a procedure within a batch file. Functionally, a procedure corresponds to a subroutine in Atari BASIC or a procedure in Turbo-BASIC XL. The definition of a procedure begins with 'PROC', and ends with 'RETURN', in the following manner:

```
PROC name
...
RETURN
```

Such a procedure can be called with:

```
GOSUB name
```

Alternatively, 'GOSUB' can also call ordinary GOTO-labels (see above), provided the command sequence marked so is ended with 'RETURN'.

Procedure calls can be nested, the maximum number of nested calls ever possible is 20. This number however can be limited down by other factors, so it is not recommended to exceed 8 levels of nesting.

### 5.2.9 Other Batch File Commands

The 'EXIT' command causes an immediate termination of the batch file processing. Alternatively you can add an exit code to be interpreted by the system as an error code. For example, 'EXIT 170' causes the batch file to be terminated with 'File not found' error. The 'SETERRNO' command causes the 'ERRNO' system variable to be overwritten with the given value. For example:

```
SETERRNO 170
```

will set 170 as the last-occurred error in the system. The keyword's most obvious usage is to clear the variable before execution of a command, that is about to be controlled later with the IF ERROR conditional.

**Technical Remark:** While a batch file is being executed, the Command Processor is periodically loaded to the memory and unloaded (it is not a resident program). This has some negative influence on interpretation speed. To speedup this process, the Command Processor can be held in the memory whilst the batch file is being interpreted. You can accomplish that by adding the command 'LOAD COMMAND.COM' at the beginning of your batch file. Before the batch file exits, the Command Processor can be unloaded with LOAD alone.

### 5.3 I/O Redirection

SDX implements I/O redirection in a totally different manner than any other DOS on A8. The command line is used to redirect the standard input (stdin) and output (stdout).

Now it is possible to divert the output of a single command by including '>>d:fname' on the command line. Similarly, input redirection is accomplished by including a '<<d:fname' on the command line. Some examples:

DIR >>PRN               redirects the output from the DIR command directly to the printer.

TYPE fname >>dest       copies a file, but will be slower than the copy command and will not preserve the time/date stamp.

BASIC <<AUTOGO       will run the BASIC program 'START.BAS', if the text file 'AUTOGO' contains the line: RUN "D:START.BAS"

The following batch file will allow paged viewing of the output of any program by redirecting it to a temporary file, then TYPEing the temporary file with the pause option:

```
%1 %2 %3 %4 %5 %6 %7 %8 %9 >>TEMP
TYPE TEMP /P
PAUSE
DEL TEMP
```

Name the above batch file MORE.BAT and type '-MORE CAR:ARC.COM' to read the directions printed by the ARC program in paged view.

## 5.4 Pipes

SDX as of version 4.42 features the mechanism of pipes, known from MS-DOS 5.0. It can be seen as extension to the I/O redirection described before. The difference is, that while in the I/O redirection the program's output can be redirected to a file and the program's input can be fed from a file, the pipe allows to send data directly from one program to another. You do not need to trouble yourself with a temporary file.

Example: To read the directions printed by the ARC program, one method is to redirect its output to a temporary file and use TYPE or MORE to view it:

```
D1:ARC >>INFO.TXT
D1:MORE <<INFO.TXT
D1:DEL INFO.TXT
```

The pipe allows to send data directly from the ARC program to the MORE command and the user does not have to care about the temporary file by typing

```
ARC | MORE
```

The MORE command is somehow exceptional, because it expects the input from the CON: device, i. e. the screen editor. Most SDX utilities and commands expect the input from a regular disk file. This requires the name of such a file to be given as a command line argument. The TYPE command can be an example here. The special name '-', for which the current pipe stream is understood, is reserved for such occasions.

Another example: if you prefer TYPE over MORE, the command line will look like this

```
ARC | TYPE - /P
```

The pipe can connect up to nine commands. For example, the external INVERSE command from the SDXTK displays the given text in inverse video.

The command line will look like this:

```
ARC | INVERSE | TYPE - /P
```

**Note:** The pipe-lining mechanism only works correctly when the variable \$TEMP points to a valid directory on your disk. The variable is normally set up by the ramdisk driver (RAMDISK.SYS) when it is loaded. If you do not use RAMDISK.SYS, you will need to set up the variable manually in your custom CONFIG.SYS file. See chapter 8 for further details.

## 5.5 Search Path

Whenever a command is given to the command processor without a drive and/or path being specified, a check is made to see if it is an internal command (such as ERASE). If not, the list of installed external commands (such as TD or KEY after they have been run once) is searched. If the command is not found, then a check is made to see if the environment variable \$PATH exists. If it does, all of the devices and/or paths named in



the variable are checked for the command in the order specified. If the command is still not found, the default directory is searched.

The \$PATH variable provides a high degree of flexibility and power, allowing to keep often used utilities out of the way in subdirectories. This is particularly useful if running a hard drive, since the main directory can get very cluttered.

The search path is used as well for batch files, the X command, the BASIC command and the CAR command, helping to find whatever the user wants them to access.

The \$PATH variable can be examined by typing PATH with no parameters at the CP prompt. The default \$PATH is 'CAR:'.

The default search path may be either changed by typing 'PATH path1;path2;...;pathn' at the command line or by using the SET command in a configuration file or batch file when booting the system, and it may be temporarily set by the APPEND command.

Each device and/or path specified must be separated from the others with a semicolon (;). It is a good idea to leave CAR: as the first entry, since many often used commands are in this device and can be called much more rapidly than the others. Order is important, since the entries will be searched one after the other.

E. g. **PATH CAR::A:\DOS\;A:\TOOLKIT\;D9:>;A:>;>** will search the CAR: device, the directory DOS on D1:, the directory TOOLKIT on D1:, the main directory of D9: (could be a ramdisk), the main directory of D1:, the main directory of the default drive. The current directory of the current drive is always scanned by SDX, so there is no need to include it into the search path.

If a new path is specified on the command line, it will replace the path currently in use and all future searches will be executed following the new path.

The search path is also used for batch files, the X command, the BASIC command and the CAR command. This is quite convenient and keeps the main directory clean from a lot of files. SDX will find them in their home directory following the set path.

It is also possible to access the search path when opening a file for read only from BASIC or any other language by adding 32 to the AUX1 value of the OPEN command.

For example **OPEN #1,4+32,0,"D:CONFIG.DAT"** will search the path defined by the environment variable \$PATH for the file. This will not work when opening a file for write or update, since this could cause unexpected and possibly dangerous things to happen.

## 5.6 Automatic evaluation of environment variables

As of SDX 4.42 the Command Processor can automatically evaluate and substitute the values of environment variables typed by the user in the command line. To invoke this function, you need to precede the variable name with the '\$'-sign.

E. g. the command 'ECHO \$PROMPT' will display the value of the variable \$PROMPT as found in the environment. If there is no such variable, the substitution will not take place - i. e., for example, if the Command Processor cannot find the \$PROMPT in the environment, the command above will display '\$PROMPT' on the screen. This principle,

that somehow differs from the behavior expected on MS-DOS or UNIX, has been established to avoid problems with PEEK and POKE commands, which can accept hex arguments, preceded with the '\$'-sign.

This mechanism can be switched off for the '\$'-strings by preceding them with the backslash. For example 'ECHO \ \$PROMPT' will always display '\$PROMPT', and never the variable's value.

## 6 Programming with SpartaDOS X

### 6.1 SDX Functions from BASIC

Many SDX features may be accessed in BASIC, ACTION!, machine language and other programming environments. The following is a list of common BASIC functions and XIO statements that allow the programmer to accomplish a variety of tasks. Conversion to other languages should not be difficult; please refer to the specific language manual for details.

In this list, 'IOCB' refers to an Input/Output Control Block (channel) number from 0 to 7. 'IOCB #0' is used by the Atari OS for the screen editor, so it should not be used.

An 'AtariDOS disk' is one initialized in DOS 2 format, whether in single, medium or double density, as produced by AtariDOS 2.0S and 2.5, MyDOS, other DOS 2 clones, and the SDX Formatter when used in this mode.

'd:', 'path', and 'fname.ext' refer to any legal SDX device identifier, pathname, and filename with extension as defined in chapter 4.

### 6.2 Notes on the Default Drive

Please remember that D: from BASIC or another language refers to the default drive, not necessarily drive #1. With SDX from the command processor D: refers to drive #4. With most other DOS types including earlier versions of SD, D: represents D1:.

```
OPEN #1,4,0,"D:TEST.TXT"
```

will open the file TEST.TXT on the default drive, not necessarily drive #1, for read under SDX.

### 6.3 Accessing the 'Kernel' Through CIO

The D: device available through the CIO with SDX is not just the disk drive handler; it is the handler for the SD 'kernel'. Any 'Kernel' device may be accessed through the CIO from any application by preceding its name with D. For example,

```
OPEN #3,8,0,"DPRN:"
```

will open the printer for output.

This also means that D4:, DD:, DD4:, DDD:, DDSK4:, and DDSKD: all refer to drive #4. When referring to a device other than a disk drive or the CAR: device, the fname.ext part of the syntax is ignored. If this confuses you just ignore it and use D1: - D9:, E:, P:, R:, and so on as you would with any other DOS. Remember that drives J: to O: can only be accessed by letters, e. g. 'DO:' for drive 15.

### 6.3.1 Open File

---

#### Purpose

Open a disk file for access through SDX.

#### Syntax

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

#### Remarks

This command opens a disk file through SDX. Aux1 is the mode (output, input, update, directory, etc.) in which the file will be opened. The following is a list of legal values for aux1. Unless otherwise noted, aux2 should be 0.

- 4 Open the file in read only mode.
- 6 Open a formatted directory. Provides a directory listing as do the DIR and DIRS commands from the command processor. **Aux2** is used to determine the style of the directory. If **aux2** is 0, standard DIRS format will be used. If **aux2** is 128, the long DIR format, including size in bytes, date, and time, will be used.
- 8 Open the file in write only mode.
- 9 Open the file in append mode. Data will be written to the end of an existing file. If the file does not exist it will be created.
- 12 Open the file in update mode. Allows reading from and writing to a file.

**Note:** On a SD format disk it is possible to position and/or write past the end of a file while in update mode.

#### An Example

This short BASIC program will read the formatted directory of a disk in drive #1 in long format and print it to the screen:

```
10 DIM ENTRY$(64)
20 OPEN #1,6,128,"D1:*.*)"
30 REM The TRAP will cause the program to jump to line 80
40 REM when the end of the directory is reached.
50 TRAP 80
60 INPUT #1,ENTRY$:PRINT ENTRY$
70 GOTO 60
80 CLOSE #1
```

### 6.3.1.1 Directory formatting attributes

As of SDX version 4.42 opening the formatted directory input with BASIC's OPEN instruction provides the following options. When aux1 is equal to 6, the bits in aux2 enable these functions:

Bit	Value	Description
7	+128	long directory format
6	+64	display attributes (p, a, h)
5	+32	put a space between filename and extension
4	+16	place a dot instead of that space (only when +32)
3	+8	long directory: suppress seconds short directory: directory extensions, ':' if +2
2	+4	24-hour time format
1	+2	two spaces before filename, '*' for protected files
0	+1	long directory: full length of the file (10 digits) short directory: file length in sectors

Bit 2 (+4) set selects the 24-hour clock when the US time format is selected globally, i. e. when the \$DAYTIME is 1. This bit is ignored, when the global time format is the EU format.

The operation of bits 0 and 3 depend on the state of the bit 7. Bit 3 set to 1 suppresses displaying the seconds of the time stamp. Bit 0 set to 1 causes the file length to be displayed in bytes as a 10-digit number. If this value exceeds 999999 bytes the length is displayed in kilobytes.

When bit 7 is off, setting bit 3 causes directories to be marked with the ':' in front of the name and displaying the directory extensions (or DIR in inverse video otherwise). Setting bit 0 causes the file's length to be displayed (in sectors) or to be suppressed otherwise.

To maintain backward compatibility, a 128 selected by the user is internally translated to 168 (\$A8, 128+32+8) and a 0 - to 11 (\$0B, 8+2+1).

To maintain backward compatibility, a 128 selected by the user is internally translated to 168 (\$A8, 128+32+8) and a 0 - to 11 (\$0B, 8+2+1).

**Note:** These 'compatible' values will never return more than 40 characters per line of the formatted directory. But selecting some other formatting attributes may cause the line to be longer than 40 characters - up to 64 characters. This is the amount to reserve for the records to be read.

Currently, the longest line is 49 characters (48 plus an EOL character). Such listing is returned after 227 (\$E3, 128+64+32+2+1) is given as an aux2 value to the OPEN call. The directory entry line may look as follows:

```
BACKUP12 TAR ... 10522112 2-19-08 3:36:34p
```

As of SDX 4.41 the short directory format (compatible to AtariDOS 2) has been slightly changed. Earlier versions displayed no directory extensions but the text 'DIR' in inverse video instead. In the current version the directory extensions are always displayed in

short directory format and the colon sign ':' is put in front of the directory filename. This format is compatible to MyDOS.

### 6.3.1.2 Accessing the Raw Directory

Setting bit 4 of **aux1** (i. e. adding 16 to 4, 8 or 12 for read, write or read/write mode) puts the OPEN in raw or unformatted directory mode. This allows to you read from and/or write to SD directories as if they were normal data files. Although this is much faster than reading a formatted directory, there is no easier way to trash a disk and make it unusable than to make a mistake in the raw directory. Unless you feel confident about what you are doing and are using a disk you don't mind losing, stay away from the raw directories! This mode will work with AtariDOS disks if the ATARIDOS.SYS driver is installed. The driver translates the Atari directory format into SD format and back.

In scan mode there is no possibility to select various formatting attributes as described above; only standard long or standard short listing is available.

### 6.3.1.3 Scan Mode

Adding 64 to **aux1** will place the OPEN in attribute scan mode. **Aux2** is used to determine the desired attribute. If a long directory is wanted in scan mode, then 128 should be added to **aux1** instead of **aux2**.

To determine the file attributes to be scanned, the following values should be added to **aux2**, assuming an initial value of 0:

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

Only those files that fit the requested description will be referenced. A value of 0 in **aux2** will ignore all attributes, even 'Hidden'.

For example, to get a long format directory of only the hidden files in the BASIC example on top of this page, simply substitute the following line:

```
20 OPEN #1,6+64+128,2,"D1:*.*)"
```

For a short directory listing without subdirectories, use

```
20 OPEN #1,6+64,128,"D1:*.*)"
```

and, finally, for a long listing of the unhidden, protected entries that end with a .COM extender, use

```
20 OPEN #1,6+64+128,1+32,"D1:*.COM"
```

It is possible to select contradictory conditions (such as 1+16, protected and not protected) for each of the attributes. This will not produce an error but, since no directory entry can match both conditions, will always select no files.

### 6.3.2 Rename File(s) (RENAME)

---

**Purpose**

Change the name of a file or a group of files.

**Syntax**

```
XIO 32,#IOCB,0,X,"Dd:[path]fname1.ext fname2.ext"
```

**Remarks**

The name of the file or names of the files specified by fname1.ext will be changed to fname2.ext, exactly as with the RENAME command from the command processor. The selected IOCB should be closed preceding this operation. Wildcards may be used in both file name specifications.

As of SDX 4.42, this call features the following extension: if X is 0, this function renames regular files, as it was before. If X is 128, this is RENDIR - and it renames directories.

**Note:** SDX 4.2x allowed to give the same name to multiple files in a directory. To avoid hassles new routines have been implemented with SDX 4.4x making this impossible.

If you receive disks having this problem from other users, it will be impossible to refer to one file without referring to the other(s). For a few verbose ways to recover from duplicate file names, refer to the RENAME command remarks in chapter 4.

### 6.3.3 Erase File(s) (ERASE)

---

**Purpose**

Remove files from a disk.

**Syntax**

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

**Remarks**

The file or files specified will be erased from the disk. The selected IOCB should be closed preceding this operation. Wildcards may be used. While it is possible to recover erased files in some instances (see the command UNERASE in chapter 4), it is wise to be very careful with this command.

### 6.3.4 Protect File(s) (ATR +P)

---

**Purpose**

Prevent a file or several files from being changed or erased.

**Syntax**

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

**Remarks**

This will allow the specified files to be opened in read mode only. Wildcards may be used. The selected IOCB should be closed preceding this operation. Protected files may not be erased, changed, overwritten, or renamed.

### 6.3.5 Unprotect File(s) (ATR -P)

---

#### Purpose

To allow files previously protected to be changed or erased.

#### Syntax

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

#### Remarks

This removes the protected status of files previously protected with the ATR command or the Protect Files XIO command. The file or files may now be erased, renamed or changed. Wildcards may be used. The selected IOCB should be closed preceding this operation.

### 6.3.6 Set File Position - POINT

---

#### Purpose

Direct access to specific points within a file and even past its end, if necessary.

#### Syntax

```
X=POS
Y=0      (see text)
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:"
```

#### Remarks

Unlike AtariDOS and compatibles, that use an absolute physical disk position (sector and offset into sector) for the NOTE and POINT functions, SDX uses a relative position within the file. POS is the desired offset into the currently open file. For example, if POS was 612, the next GET from the file would get the 613th byte of the file. This value will refer to the same position in the file even if the file is physically moved to another medium. The file must be open for this operation.

Because of a limitation in Atari BASIC, BASIC XL, and BASIC XE, the first method using the POINT command will only work with positions up to and including 32767. If a value greater than 32767 is given a BASIC error will occur. To POINT to a location with a larger number with these languages (and possibly others) it is necessary to use the second method.

The POINT command is bypassed by poking the three byte file position directly into the IOCB registers and executing the XIO. **Aux1** and **aux2** must be the same values used when the file was opened.



Other languages, such as ACTION! and Turbo-BASIC XL have no such limitation on the POINT command, allowing it to be used instead of the lengthy XIO method. In this case, use the following format:

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

If you are an user of an earlier version of SD, you should notice that NOTE and POINT now work the same way with AtariDOS disks as they do with SD disks. POINT will not use sector number and offset regardless of the disk format.

Using NOTE and POINT with SDX on an AtariDOS disk may prove to be time consuming. Since determining the relative offset into the file needs to be read from the start every time a POINT is used. This also causes segmented binary files to take much longer to load from AtariDOS media than from SD disks. NOTE and POINT tables created by other DOS types (including earlier versions of SD) for files on AtariDOS disks will no longer be valid.

### **Sparse Files**

On a SD medium, it is possible to point past the end of a file opened in append mode. When data is placed in a file past the end, the file is given the new length, but no physical sectors are used for the space between the old and the new data. In the sector map of the file, the unallocated sectors are represented by a sector number of 0. Should you at any time write to a position in this gap, a sector will be allocated. This gap may not be read, and a file containing gaps may not be copied. An error will occur if either is attempted.

**Note:** The internal SDX routine, corresponding to this function, has been rewritten, so that random access to very long files (greater than 500 KiB) should now work up to four times faster than on SDX 4.2x.

### **6.3.7 Get Current File Position - NOTE**

---

#### **Purpose**

Determine the current position within a file.

#### **Syntax**

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

#### **Remarks**

This will return the relative current position within the currently open file; i. e., the offset into the current file. This will not return sector number and offset into the sector, regardless of disk format. The file must be opened for this operation. For users of SD 2.x and 3.x, you may be interested to learn that this method works with those versions. The XIO 38 command described in the SD Construction Set manual will still work but is redundant.

### 6.3.8 Get File Length

---

#### Purpose

Determine the length of the currently open file.

#### Syntax

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

#### Remarks

This will return the length of the currently open file. IOCB, aux1, and aux2 should be the same values used for the preceding opening of the file.

### 6.3.9 Load a Binary File (LOAD)

---

#### Purpose

Load and execute a binary file from another program.

#### Syntax

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

#### Remarks

This command will load a binary file and handle it according to the following loading modes for X:

- 0 Load the program using INIT and RUN vectors (\$2E2, \$2E0). If the program does not set the RUN vector, it is executed by a jump to its first address. This is SDX default.
- 64 Use INIT and RUN vectors only (AtariDOS compatible mode).
- 128 Do not execute. No INIT/RUN vectors are used.
- 192 Do not execute, but perform INITs.

Note that previous SDX versions performed INITs regardless of the settings.

The IOCB should be closed. Loading a binary file from an AtariDOS disk will take much longer than loading the same file from a SD format disk.

### 6.3.10 Change Default Drive & Directory

---

#### Purpose

Change default drive & directory at the same time.

#### Syntax

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

**Remarks**

This will change the current drive & directory. When the new drive is accessed without reference to a specific directory, the default one will be used. The rules regarding path and IOCB status defined in XIO 42 apply here.

**6.3.11 Create a Directory (MKDIR)**

---

**Purpose**

Create a new subdirectory.

**Syntax**

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

**Remarks**

The directory 'newdir' is the directory that will be created. Any path before this must be valid. For example, if

```
XIO 42,#1,0,0,"D1:LARRY>MOE>CURLY>SHEMP"
```

is used, then the path 'LARRY>MOE>CURLY>' must already exist from the current directory, for the directory 'SHEMP' to be created.

The selected IOCB should be closed preceding this operation. This will ONLY work for SD formatted disks.

**6.3.12 Delete a Directory (RMDIR)**

---

**Purpose**

Remove an existing directory.

**Syntax**

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

**Remarks**

The directory olddir will be deleted. A directory must be empty to be deleted. The rules regarding path and IOCB status defined in XIO 42 apply here.

**6.3.13 Change Current Directory (CHDIR)**

---

**Purpose**

Change the current working directory of a disk.

**Syntax**

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

**Remarks**

This will change the directory that is used when the specified drive is accessed without reference to a specific directory. The rules regarding path and IOCB status defined in XIO 42 apply here.

### 6.3.14 Set Boot File (BOOT)

---

#### Purpose

Establish the file that will be loaded when the computer is initialized without the use of SDX.

#### Syntax

```
XIO 45,#IOCB,0,0,"Dd:[path]fname.ext"
```

#### Remarks

This will cause the specified file to load when the computer is turned on or cold started and the SDX cartridge is not used. With earlier versions of SD, the primary use of this was to cause the \*.DOS file to be booted. With SDX, the uses of this command are limited. The IOCB should be closed and a SDFS disk must be used.

**Note:** BOOT will not work with all binary files. There are many specific rules that must be followed when loading a file without DOS. The primary purpose of this command is to load a DOS module.

### 6.3.15 Set Attributes (ATR)

---

#### Purpose

Manipulate the protected, hidden and archived status of files.

#### Syntax

```
XIO 49,#IOCB,aux1,aux2,"Dd:fname.ext"
```

#### Remarks

Used to modify attributes of a file. Wildcards are allowed in fname.ext. Aux1 is used to select the attributes to be changed and whether they will be set or cleared. Aux2 is used to determine the files to be affected. To perform the desired attribute modification, add the following values to aux1, assuming an initial value of zero:

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

Aux2 is used exactly as with the scan mode of the OPEN statement. It will select the files to be affected by current attribute status. These values should be added to aux2, starting with a base value of 0:

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

For example, to hide all of the files on drive #1 with a .BAK extender, use

```
XIO 49,#1,2,0,"D1:*.BAK"
```

To protect and set the archive bit for all of the hidden files on drive #1, use

```
XIO 49,#1,1+4,2,"D1:*.*"
```

and to unhide and unprotect all the hidden files with a \*.BAK extender on drive #1, use

```
XIO 49,#1,16+32,2,"D1:*.*BAK"
```

The selected IOCB should be closed preceding this operation.

### 6.3.16 Format a Disk (FORMAT)

---

#### Purpose

Initialize a disk, setting up the appropriate track, sector, and directory data.

#### Syntax

```
XIO 254,#IOCB,0,0,"Dd:"
```

#### Remarks

The Dd: specified is irrelevant, since this command will bring up the SDX disk formatter menu. From this menu, disk number, format, size, skew, etc. may be selected. <ESC> will exit the program and return control to the previous environment. This allows media of all types to be formatted from within any program. The selected IOCB should be closed preceding this operation.

**WARNING!** Hide and protect will not save files from being destroyed by formatting a medium. For details please refer to the format command in chapter 4.

**Note:** The next two commands are not available through XIO calls. They must be accessed directly through the CIO. An assembly language listing of a routine to access these will follow, along with a BASIC program that demonstrates its use.

### 6.3.17 Get Disk Information (CHKDSK)

---

#### Purpose

Read information about a disk

The next two paragraphs contain information from SDX 4.20 to help understand the differences to the changed version:

#### CIO Data

```
iccom = 47
icbal = low byte of 'Dd:' address
icbah = high byte of 'Dd:' address
icbll = low byte of buffer address
icblh = high byte of buffer address
```

### CIO Output Results

buffer = results of CHKDSK operation (17 bytes)  
 +0 = version number of disk, 0 if AtariDOS format  
 +1 = number of bytes per sector, 0 if 256  
 +2 = total number of sectors on disk (2 bytes)  
 +4 = Number of free sectors on disk (2 bytes)  
 +6 = volume name, always 'AtariDOS' for AtariDOS format disks (8 bytes)  
 +14 = volume sequence number, 0 if AtariDOS format  
 +15 = volume random number, 0 if AtariDOS format  
 +16 = unused byte

### Changes to this function implemented with SDX 4.4x

The earlier versions of SD support disks with 128- and 256-byte sectors. With SDX 4.4x support is added for 512-byte sectors and theoretically even larger.

The CHKDSK function returns the 17 bytes described in before. The byte holding the information about the sector size, found at the offset buffer+1, has a value of 128 for the 128 bytes per sector densities (Single, Medium) and 0 for 256 bytes per sector (Double). Originally it is just the low byte of the actual sector size value (128=\$0080, 256=\$0100) - encoding of sector sizes larger than 256 bytes is impossible this way.

Therefore the interpretation of this byte has changed in SDX 4.4x. Both inside the DOS itself and in its external utilities. The new way is of course backward compatible, the 'old' values still retain their traditional meaning.

The number 128 still signifies the number of bytes per sector. Any other value is the high byte of the logical sector size value in bytes, minus 1. This allows for easy encoding of sector sizes from 128 bytes to 64 kilobytes:

Size (BPS)	Hex Value	Encoded As	Remarks
128	\$0080	\$80 (128)	*
256	\$0100	\$00 (0)	*
512	\$0200	\$01 (1)	
*1024	\$0400	\$03 (3)	**
*2048	\$0800	\$07 (7)	**
*4096	\$1000	\$0F (15)	**
*8192	\$2000	\$1F (31)	**
*16384	\$4000	\$3F (63)	**
*32768	\$8000	\$7F (127)	**
*65536	\$0000	\$FF (255)	**

\* = backward compatibility to SDX 4.1x and 4.2x.

\*\* = not supported yet)

An assembly routine that calculates the real sector size value out of the 'encoded' one can look like this:

```
; the input code is given in the
; accumulator (A), the result
; is returned in A (low byte)
; and X (high byte)
```

```

;
getssize      ldx #$00
              cmp #$80
              beq quit
              tax
              inx
              lda #$00
quit          rts

```

### 6.3.18 Get Current Directory Path (CHDIR)

---

#### Purpose

Get the path from the root directory to the current directory of a drive

#### CIO Data

```

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

```

#### An Example

The following is a short BASIC program demonstrating the use of the last two CIO calls. Next is an assembly listing of the code contained in the DATA statements and the string CIO\$.

```

10 DIM CIO$(32),BUFFER$(64),DRIVE$(4),CHKDSK(17)
20 DRIVES="D1: ":DRIVE$(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)=BUFFERS
110 ?:"CIO Call Demonstrator"
120 ?:"1 -> CHKDSK"
130 ?:"2 -> Path to current directory"
140 INPUT CHOICE
150 IF CHOICE<>1 AND CHOICE<>2 THEN GOTO 120
160 ICCOM=CHOICE+46
170 ?:"Which drive";:INPUT D
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)=">":

```

```

        BUFFERS=BUFFER$(1,X):POP:GOTO 260
250 NEXT X
260 ? BUFFER$:GOTO 80
270 FOR X=1 TO 17:Y=ASC(BUFFER$(X,X)):CHKDSK(X-1)=Y:NEXT X
280 ?"Volume: "; BUFFER$(7,14)
290 ?"Bytes/sector: ";
300 IF CHKDSK(1)<>128 THEN CHKDSK(1)=(CHKDSK(1)+1)*256
310 ? CHKDSK(1)
320 ?" Total bytes: ":
330 ? CHKDSK(1)*(CHKDSK(2)+256*CHKDSK(3))
340 ?" Bytes free: ":
350 ? CHKDSK(1)*(CHKDSK(4)+256*CHKDSK(5))
360 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

```



## 6.4 SpartaDOS User Accessible Data Table

---

Several SD variables are available to programmers to allow easy access to the command line for applications and utilities. This data table is referred to as COMTAB and is pointed to by the OS variable DOSVEC at memory location 10 (\$0A). An assembly language example is included for aid. This table is valid with all versions of SD except where noted. Locations COMTAB, ZCRNAME, BUFOFF, COMFNAM, and LBUF are also supported by OS/A+ and DOS XL.

As of SDX V. 4.48 U\_GETADR can parse expressions such as 'address+constant', 'address-constant', where the 'address' part is a numeric address value or a symbol, and the 'constant' part is a numeric constant.

Therefore the utilities which use this call (BLOAD, DPOKE, PEEK, POKE, MDUMP) will now allow syntax like: 'PEEK COMTAB2-4' or 'POKE 1536+32,1'.

The constant is only added to the lower 16 bits of the address, thus when the result exceeds 65535, the low word will wrap back to 0.

### **DECOUT2            COMTAB-21**

Contains the right-justified, space-padded output of the misc\_conv32 routine, an ASCII string representation of the four byte number at DIVEND (see Page Seven 'Kernel' Values). 10 bytes (including 8 byte DECOUT).

### **DECOUT            COMTAB-19**

SDX only. Contains the right justified, space padded output of the 'misc\_convdc' routine, an ASCII string representation of the three byte number at DIVEND (see Page Seven 'Kernel' Values). (8 bytes)

### **LSIO                COMTAB-10**

A pointer to the SD high speed SIO routine. You can use the address contained here instead of \$E459, the OS SIOV, to perform high speed sector I/O with your programs.

### **DIVEND            COMTAB-6**

SDX only. A three byte number here will be converted by the 'misc\_convdc' routine to a string at DECOUT (→ Page Seven 'Kernel' Values). A four byte number here will be converted by the misc\_conv32 routine to a string at DECOUT2 (→ Page Seven 'Kernel' Values).

### **WRTCMD            COMTAB-2**

This location contains the SIO write command. A 'W' here indicates write with verify, while a 'P' indicates write without verify.

### **COMTAB            COMTAB+0**

This is a 6502 jump instruction followed by the address of the DOS entry routine. A jump here enters DOS.

### **ZCRNAME            COMTAB+3**

This is a 6502 jump instruction followed by the address of the file name crunch routine. This location is used to interpret the command line. A jump here will pull the next command from LBUF, translate the drive or device identifier if one is given (i. e., A: to D1:), add the default drive identifier if none is given and place the result at COMFNAM. Each

call will advance BUFOFF to point to the next entry on the command line, so that each call to the crunch routine will get the next entry on the line. If no entries remain, the 6502 zero flag will be SET on return. Since the 6502 has no indirect jsr, it is necessary to use a few lines of code to access this routine. An example will follow this list.

**BUFOFF            COMTAB+10**

The offset into LBUF where the next parameter to be read is located. This can be manipulated to reread the command line.

**DATER            COMTAB+13**

The date in DD/MM/YY format (3 bytes). Updated by VGETTD. Updated continuously while the Time/Date line is on with SDX.

**TIMER            COMTAB+16**

The time in HH/MM/SS format (3 bytes). Updated by VGETTD. Updated continuously while the Time/Date line is on.

**ODATER           COMTAB+19 (3 bytes )**

**OTIMER           COMTAB+22 (3 bytes )**

**TDOVER           COMTAB+25 (1 byte )**

The function of these registers is similar to the function of DATE, TIME and DATESET registers respectively, except the TDOVER not being automatically cleared after use (unlike DATESET).

ODATER, OTIMER and TDOVER are the old date/time stamp control registers used with SD 3.x. SDX 4.2x disabled and replaced them with the new triad of DATE, TIME and DATESET. This was the reason, why SD 3.2 copy programs under SDX 4.2x could not control timestamps in the files copied.

SDX 4.4x restores the function of these registers. DATESET still has the highest priority and when it is set, the TDOVER value is ignored.

After a program terminates in SDX 4.4x, TDOVER is cleared. This is not the case in version 3.2. Thus it is good programming practice to clear this register when the program is about to quit to DOS (if TDOVER was used).

**\_800FLG           COMTAB+27**

SDX only. \$FF if the computer is an Atari 800. Zero otherwise.

**NBANKS           COMTAB+29**

SDX only. The number of free expansion memory banks. This is the number also shown by the MEM command.

**BANKFLG           COMTAB+30**

SDX only. \$FF if USEing BANKED. Zero otherwise.

**OSRMFLG           COMTAB+31**

SDX only \$FF if USEing OSRAM. Zero otherwise. **Note:** USE NONE is indicated by both BANKFLG and OSRMFLG being zero.

**COMFNAM           COMTAB+33**

This is the destination buffer for the ZCRNAME routine. It will always begin with a Dd: since the default drive is added if none is given. If you are looking for switches or other options, start looking at COMFNAM+3. This buffer is 28 bytes long. Therefore the first parameter on a command line may not exceed 25 characters.

### **LBUF                   COMTAB+63**

This is the input buffer for the command processor. The entire command line is stored here. LBUF is 64 bytes long.

### **COPYBUF               COMTAB+191**

This is the main buffer used by the SDX 'kernel'.

### **An Example**

The following assembly language program demonstrates one way to read the SD command line. It simply echoes the command line with the drive specifications added or translated as necessary. The name of the command itself is not printed.

```

;C10 and IOCB equates
ciov = $E456
icom = $0342
icbal = $0344
icbah = $0345
icbll = $0348
icblh = $0349
write = $09

; SpartaDOS equates
comtab = 10
zcrname = 3
bufoff = 10
comfnam = 33

; The program.

      *=$4000           ; or wherever.
                        ; patches our crunch routine to
init   ldy #zcrname+2  ; be the same as the COMTAB one.
      ldx #2
loop1  lda (comtab),y
      sta crunch,x
      dey
      dex
      bpl loop1

mainloop
      jsr crunch       ; get next command line entry.
      beq exit        ; quit if there are no more.

; Set up for C10 print of data at COMFNAM

```

```

        ldx #0           ;IOCB #0 (E:)
        lda #63         ; set buffer length for max
        sta icbll,x
        lda #0
        sta icblh,x
        lda comtab      ; store COMTAB+33 at icba
        clc
        adc #comfnam
        sta icbal,x
        lda comtab+1
        adc #0
        sta icbah,x
        lda #write      ; 'print string' command
        sta iccom,x
        jsr ciov        ; print it.
        jmp mainloop
exit    rts
crunch jmp $FFFF      ; will be changed by INIT routine
        *= $02E0
        .word init

```

## 6.5 Decoding the drive identifier

---

SDX 4.4x supports fifteen drive identifiers. D1: to D9: or DA: to DI:, as already known from SDX 4.2x, and drives DJ: to DO: (letters only).

The convenience known from the previous SDX X versions, namely that the drive 1 can be referenced either as D1: or DA:, drive 2 as D2: or DB: and so on has been kept of course.

When a program receives a drive identifier from the DOS (e. g. as a command line input) and passes it on to another DOS function unchanged, there should be no problems with the new, 'non-standard' drive identifiers. There will be no difference in the code either, when a program wants to calculate the real (binary) drive number - for DUNIT for example. To do that, it is enough to clear the high nibble of the drive digit or letter with an AND #\$0F.

A reverse conversion may be a problem though, because a \$30 should be added to a DUNIT value to get a drive digit, or a \$40 to get a drive letter. It was not necessary to distinguish these two cases so far, so the programs doing such a conversion on purpose will probably not work correctly on drives 10-15. Luckily such a calculation is rarely necessary, but we supply an example just in case:

```

dsk2asc
        lda dunit
        ora #$30
        cmp #'9+1
        bcc ok
        adc #$0f
ok      rts

```

The result - an ASCII character symbolizing the given drive - is returned in the accumulator. It will work with any DOS. But if the program is to be used with SDX only, the routine may be greatly simplified:

```
dsk2asc
    lda dunit
    ora #$40
    rts
```

## 6.6 Symbols

---

A symbol is an eight-character name of an object residing somewhere in the computer's memory. Such an object can be a routine or a data structure. When the symbol name is known to the programmer, it can be translated inside the program to the actual address. Normal binary programs have to do that 'by hand', i. e. making the appropriate system call (see Page 7 'Kernel' Values). This is neither the most convenient nor the only way to do that; some assemblers can generate SD specific, specially structured binaries; in case of such a binary the symbol-to-address translation is done automatically by the loader.

If a symbol exists, that means that the corresponding routine is loaded into the memory and the symbol itself provides the information where it can be found. Addresses pointed to by symbols can change depending on the SD version or the order of loading drivers. A part of the symbols points to the ROM, part of them is even defined in ROM, but even in such a case they cannot be considered fixed forever - every symbol can, at any moment, get replaced by its new instance pointing to another place in the memory. This happens when a device driver replaces or patches a system procedure.

If a symbol does not exist, most of the time it means that the corresponding driver is not loaded to the memory.

## 6.7 Vectors Under the OS ROM

---

The vectors are created under the OS ROM to secure some backward compatibility with SD 3.2 and earlier versions. There is no need to use them in SDX, as the same routines are pointed to by appropriate symbols. So you do not need to look under the OS ROM or worry if the vectors still exist or were destroyed by a program loaded in the meanwhile (Turbo-BASIC XL, for instance).

Since these vectors are under the OS ROM, it is necessary to enable the RAM instead of the ROM in this memory area. One possible method is:

```
lda $D301    ; PIA, responsible for bank
pha         ; selecting. Store status.
and #$FE    ; RESET bit 0.
sta $D301    ; enable RAM under OS ROM.
jsr VGETTD  ; call the vector.
pla
sta $D301    ; restore PIA to previous state
```

Each of these functions contain a jump (JMP) followed by the address of the function. It is good practice to always check for this JMP instead of assuming that the vector is still there.

On the next page see alist of symbols corresponding to the old vectors:

Vector	Label	Symbol	Defined By
\$FFC0	VGETTD	I_GETTD	clock drivers (e. g. RTIME8.SYS)
\$FFC3	VSETTD	I_SETTD	as above
\$FFC6	VTDON	I_TDON	TD.COM
\$FFC9	VFMTTD	I_FMTTD	TD.COM
\$FFCC <sup>+80</sup>	VINITZ	_INITZ	kernel
\$FFCF <sup>+</sup>	VINITZ2	-/-	-/-
\$FFD2	VXCOMLI	XCOMLI	kernel
\$FFD5 <sup>+</sup>	VCOMND	-/-	-/-
\$FFD8 <sup>+</sup>	VPRINT	PRINTF	kernel
\$FFDB <sup>+</sup>	VKEYON	I_KEYON	KEY.COM

The vectors are not used by SDX itself and can be accidentally destroyed by software using the RAM under the OS ROM. So their use implies some trouble. In future versions of SDX they may disappear completely.

The symbols I\_GETTD, I\_SETTD, I\_TDON and I\_KEYON work the same way as the old vectors VGETTD, VSETTD, VTDON and VKEYON in SDX 4.2x. There is only one exception, namely that the lack of the appropriate driver being loaded is not signified by the state of the Carry-Flag after the call, but rather by the impossibility to do the call because the symbol cannot be found.

In the I\_GETTD and I\_SETTD procedures a set Carry-Flag means that the clock driver is busy at the moment. You should call the routine again. It is good programming practice to count, how many times in a row the call failed and break the loop after a certain number (e. g. 255) of iterations to avoid deadlock when the clock becomes unresponsive for any reason (e. g. a hardware failure).

The I\_FMTTD is loaded to memory along with TD.COM, being a part of it. It accepts an address of a 32-character buffer in the Y/X (low/high) registers. When the call returns with Carry-Flag set, this indicates an error (the clock driver being in permanent busy state). If the Carry-Flag is cleared, there is still time and date information in the buffer, in the form of an ASCII, EOL-terminated string.

## 6.8 Page Seven 'Kernel' Values

---

Several page seven locations allow for 'kernel' operation to be accessed. While it is beyond the scope of this guide to document all of these locations, a few may prove to be of interest.

Name	Address	Function
sparta_flag	\$700	'S' if SpartaDOS
sparta_version	\$701	version in hex; \$32 = 3.2, \$40 = 4.0, etc.
kernel	\$703	jump (JMP) to 'kernel' entry
block_io	\$706	see below
misc	\$709	jump (JMP) to 'misc' entry
device	\$761	'kernel' device number
name	\$762	filename and ext (11 bytes)

---

<sup>80</sup> The cross in the table marks the locations already obsolete in SDX 4.4x

date	\$77B	see below (3 bytes)
time	\$77E	see below (3 bytes)
dateset	\$781	see below
path	\$7A0	path only string (64 bytes)

The 'kernel' routine is called by doing a jump subroutine (JSR) to address \$703 with the desired command in the 6502 Y register and the desired device number previously stored in the memory location 'device'. For example, with a \$10 in device, a value of 100 in the Y-register will cause the current time and date to be placed in the variables 'time' and 'date'. A 101 will cause the current time to be set to the values contained in the variables 'time' and 'date'.

kd_gettd	100	get current time and date
kd_settd	101	set time and date

The block\_io vector routines support reading and writing sectors. Using this vector instead of lsis decreases the number of DCB variables to set in the program and greatly reduces worries about disk density, the combinations of the sector number and its size (in the double density, as it is widely known, the first three sectors are 128 bytes each, while the other sectors in this density are 256 bytes each).

Before placing a call to block\_io you should set the following DCB variables: the device code (DDEVIC), the device number (DUNIT), the buffer address (DBUFA) and the sector number (DAUX1/2). The function code should be passed in Y. Upon exit, the system puts the status code into the accumulator: 0 or 1 for a success, or a negative value to signify an error code.

The block\_io function codes are:

- 0 bio\_rdsec - read sector (standard, no density check)
- 1 bio\_wrsec - write sector (as above)
- 4 bio\_rdsys - check density, remember it, and read sector
- 5 bio\_sbps - remember the sector size currently set in DBYT

Other function codes (2 and 3) are reserved for internal use of the SPARTA.SYS driver.

Functions number 0 and 4 operate similarly, except that the latter fetches information about the actual density from the drive first and stores it into a memory table for later reference. The functions numbered 0 and 1 use that information and so a program that wants to access sectors this way, must always call the function number 4 first, e. g. to read sector number 1 and use the function 0 to read all other sectors.

If it is necessary to bypass the density recognition mechanism, and the sector size is otherwise known, in lieu of the function 4 one can store the required sector size to the DBYT variable (\$0308-\$0309), the required drive number to DUNIT (\$0301), call function 5 and then subsequently 0.

The following is a list of valid 'misc' vector commands. These should be loaded into the A register before executing a JSR \$709. The Y register is used as an index into COPYBUF for those operations using COPYBUF.



misc_initz	0	initialize misc driver
misc_getfina	1	convert path at COPYBUF to device, path, and name
misc_getpath	2	convert path at COPYBUF to device and path
misc_convdc	5	convert three byte number at DIVEND to a text string at DECOUT
misc_conv32	11	see following explanation

The conversion routine 'misc\_convdc' is now 32-bit, with the most significant byte of the DIVEND at COMTAB-3 and generates resulting 10-character ASCII strings at DECOUT2. But to retain the compatibility with existing applications the old misc\_convdc entry zeroes that highest byte before proceeding with the conversion, thus cutting the number down to 24 bits and the result to 8 digits. For 32-bit conversions a new entry should be used, labeled 'misc\_conv32', with 'function code 11'. This function code is available only as of SDX 4.4x. Calling it on an earlier version of the DOS will cause undefined results.

Whenever a file is opened the time and date for that file will be placed in 'time' and 'date'. When a file is opened for write only and 'dateset' equals 0, the current time and date will be read into 'time' and 'date' and assigned to the file. If 'dateset' is -1 (\$FF), the file will get the time and date that are in the variables when the open is executed. 'Dateset', unlike the old COMTAB location TDOVER from previous versions of SD, will automatically clear after use. This is how a copy of a file can retain the time and date of the original. This is also how a program like ARC assigns stored time/date information to a new file.

## 6.9 Using the CON: Drivers in Own Programs

---

**Note:** The following information is valid for the drivers bundled with SDX 4.42 or newer versions.

The screen editor provided by the CON64.SYS or CON.SYS (SDXTK) screen driver, behaves quite similarly to the standard one. You can use the same 'E:' device control codes and the same variables (such as CRSINH \$02F0) and expect compatible behavior.

**Note:** The RMARGN variable (\$53) works as expected too, with one exception: you will not be allowed to set the right margin to 39. When this happens, the console driver will reset this to 63 or 79 at the nearest opportunity.

A programmer, however, may want to take advantage of additional features of these drivers and this, for example, requires a way to detect them in memory. This section is intended to address these topics.

**Note:** for simplicity, we mark the console IOCB as '#0' throughout this section. You should remember though, that this, even though accepted by Turbo-BASIC XL, is not allowed in Atari BASIC. In Atari BASIC you have to use '#16' instead.

### 6.9.1 Detecting the extension

The following call:

```
XIO 33,#0,12,0,"E:"
```

should return a 1 (success), if either CON64.SYS or CON.SYS was loaded to the memory. An error, especially number 146 (Function not implemented) means that neither one is available.

Apart from the status, the XIO 33 call returns additional information in the ICAX3-6 bytes of the IOCB (\$034C...\$034F+IOCB\*16), as follows:

- ICAX3/4: 'direct write' entry point
- ICAX5: activity flag
- ICAX6: max. number of screen columns

The 'direct write' entry point is described below. The activity flag is 128, when the 64- or 80-column mode is currently active or 0 otherwise.

The maximum number of screen columns, regardless of the current screen mode, is 64 for CON64.SYS and 80 for CON.SYS. In other words, status 1 returned by the XIO 33 provides the information that one of these drivers is loaded and the ICAX6 - which one.

### 6.9.2 Enabling and disabling the extended mode

The ICAX6 value returned by the XIO 33 call is also the function code for the XIO call to enable and disable the extended console mode:

```
XIO nc,#0,12,m,"E:"
```

For the nc value you should substitute what XIO 33 returned in ICAX6. The m value, when it is 128, enables the extended console mode and disables it when it is 0. Adding 64 here sets the 'no force' flag - when it is set and the required mode is already active. Nothing is done (or the mode is re-enabled otherwise).

The call returns the previous value of the activity flag in ICAX3: 128, when the extended mode was active, or 0 if it was not.

### 6.9.3 The 'direct write' entry point

The 'E:' device routines do many calculations that are necessary for the screen editor to operate properly, but may be useless for the user's purpose. The side effect is that the editor is rather slow and has some limitations. For example, you cannot create a frame around the screen, because placing a character in the lower right corner of the screen will cause its contents to scroll up.

Because of that, many programs tend to bypass the editor and write data directly to the screen memory. In GRAPHICS 0 this is easy and fast. In 64-column text mode however, it is not so easy, and it is certainly not easy to make it fast, because this mode is emulated and displaying a character needs some calculations.

To facilitate this task, the driver offers a 'direct write' entry to a routine that places the character at the given x/y coordinates and does nothing more. Printing characters using that routine is about 2.5 times faster than the standard way.

**CAUTION:** The routine does no sanity checks. The burden of checking if the given coordinates fit on the screen, belongs to the calling program.

The address returned by XIO 33 in ICAX3/4 is the entry point. The ASCII code of the character to be displayed should be put into the accumulator, the x/y coordinates to the CRSCOL (\$55-56) and CRSROW (\$54) system variables, the routine should be called with JSR. It does not return any specific information to the caller.

This method is used by MENU.COM, when invoked in 64- or 80-column mode.

**Note:** The routine only displays the given character and does nothing else. For this reason, even after it was called only once, the internal variables of the 'E:' device driver may lose consistency. So, it is not advisable to mix 'direct write' calls with standard editor calls, because this may put the calling program into severe trouble. Before returning to normal editor operation (e. g. before quitting to DOS), the program should reset x and y to 0 and then send the 'Clear Screen' character (ASCII 125) to the editor in order to reset internal settings of the driver.

#### 6.9.4 Scrolling the display up

The driver provides a routine that scrolls the display contents up one line up. The call:

```
XIO 34,#0,12,n,"E:"
```

will make the screen scroll up starting from the line number 'n' and ending at the line whose number plus 1 is held by the system variable BOTSCR (\$02BF). Its normal value is 24, so XIO 34,#0,12,0,"E:" will scroll the entire display from the top (i. e. line number 0) to the bottom (i. e. BOTSCR-1, which is 23). Changing these values allows the program to scroll selected parts of the screen.

**Note:** the BOTSCR value must be greater than a zero. It may not be greater than 24 and it may not be greater than or equal to 'n'!

The text buffer and other editor variables are updated so there should be no problems with mixing standard editor calls with XIO 34. The program however, should still reset the editor in the way described above before it quits to the DOS.

#### 6.9.5 Other functions

XIO 32,#0,12,0,"E:" should be done whenever the program changes the RAMTOP value (\$64). XIO 32 will then move the Display List and the screen memory and updates the memory pointers and screen driver internal variables accordingly. Normally you never need to do that, but if you do remember that no part of the screen memory may be put into the area where memory banks are switching, i. e. \$4000-\$7FFF.



## 7 Technical Information

### 7.1 SpartaDOS File System

---

The SDFS version 2 is the standard format used by all versions of SD 2.x, 3.x SDX, BeweDOS, Micro-SD and RealDOS. With SDX 4.4x an advanced version 2.1 has been introduced. SDFS version 1.1 from SD 1.x is outdated.

There are four distinct types of sectors on a disk/medium formatted in SDFS. These are boot, bit map, sector map and data sectors. Data sectors may contain either directory data or file data. The following is a detailed discussion of each type of sector.

#### 7.1.1 Boot Sectors

As with most other DOS types for the A8, the first three sectors on the disk are boot sectors. These contain a program generally known as boot loader to load the file designated into the system when booted and other information needed to be able to store and retrieve data to and from the disk. From boot sectors formatted in 128 or 256 bytes per sector only the first 128 bytes will be used. Therefore the physical capacity of 256 byte sectors is not fully used when they are boot sectors. Many DOS or programs show them to be just 128 bytes in size. This is the reason why many users address this issue as 'boot sectors are always in single density'. For devices using 512 byte sectors this will be different.

Sector 1 from offset \$30 to offset \$7F and all of sectors 2 and 3 are the boot code that loads a file under all DOS compatible to SDFS, if specified (with the BOOT command). This code is not used with SDX. The first part of sector 1 is a data table containing the values listed below as offsets into the sector. A disk can be a floppy disk (real or emulated one), a ramdisk or a hard drive partition unless otherwise specified. All two or three byte numbers are stored in standard low byte/high byte format.

SDX 4.4x uses SDFS 2.1, which can have sectors larger than 256 bytes. Storage devices using 512 bytes per sector or even more will have the first three sectors in the same size, instead of 128 bytes per sector. Moreover, the boot region takes only one sector, the one number 1. The first 42 bytes of this sector carry information about the file system structure, just like in earlier versions of SD. The remaining portion of the sector is occupied by the new boot loader, able to handle 512-byte sectors and larger ones.

You might think that such a disk structure, which is in fact breaking the existing standard, is an unnecessary complication. Indeed, in SDX and its utilities the needed changes had to be done with regard to the mechanism of density detection. Before version 4.40, the detection process in SD was based on the known size of the first sector. It had to be 128 bytes per sector and the size of the rest of the sectors could be determined from the first sector's content. Another problem was that a 512 bytes per sector disk can be booted only as a hard drive partition, because the Atari 8-bit OS is not able to set the sector size correctly either, which for a serial drive causes a checksum error and failure.

Bottom line is there are more advantages than disadvantages with it. At first we are now compliant with the real standard disk devices used and produced around the world, where all the sectors are of the same size and the smallest possible one is 512 bytes. Secondly more boot region space is available (512 instead of 384 bytes).

Therefore creating a new boot loader was possible. Last but not least, the free space on media having 512 byte sectors is used more efficiently - although the rest of the first three sectors with less than 1.5 KiB is certainly a negligible loss on a hard disk.

The following table explains the sector 1 values given as offsets in hex (dec) into the sector. Bytes 0 to 8 are standard for an A8 boot sector and not SDFS specific. The other important bytes for the SDFS will be explained in detail as far as information is available.

- \$00 (0) Usually 0. Some formatting tools put a \$53 (= 'S) for SD here.
- \$01 (1) Number of sectors to boot. Single density max. 255, double density max. 3, double density 512 just 1.
- \$02 (2) Address where the boot sectors are loaded to. 2 bytes.
- \$04 (4) This address is copied to DOSINI. 2 bytes.
- \$06 (6) After a successful boot, the Atari-OS's boot routine passes command execution to code beginning here. In case of 128 or 256 byte boot sectors a jump to the booted code at memory location \$3080. With a 512 byte boot sector it jumps to \$0440. 3 bytes.

**Note:** these three bytes allow to identify the disk format.

- \$09 (9) Sector number of the first sector map of the MAIN directory. 2 bytes.
- \$0B (11) Total number of sectors on the disk. 2 bytes.
- \$0D (13) Number of free sectors on the disk. 2 bytes.
- \$0F (15) Number of bit map sectors on the disk.
- \$10 (16) Sector number of the first bit map sector. 2 bytes.
- \$12 (18) Sector number to begin the file data sector allocation search. This is the first sector checked when an unallocated sector is needed. This serves two purposes; it relieves the necessity of searching the bit map from the beginning every time a file is to be allocated and it allows sectors to be reserved after the main directory for directory expansion. 2 bytes.
- \$14 (20) Sector number to begin the directory data sector allocation search. This is the first sector checked when a directory is to be expanded or added. Keeping this separate from the search number above will help keep directory sectors close together to speed searches. 2 bytes.
- \$16 (22) Disk volume name. SDFS uses this as part of the disk change identification procedure. 8 bytes.
- \$1E (30) Number of tracks on the disk. If the drive is double-sided bit 7 will be set. If it is not a floppy disk (a RAM-disk or hard drive partition, e. g.) this location will contain a 1.

- \$1F (31) Size of the sectors on this disk (not boot sectors).  
 SDFS 2.0: a \$00 indicates 256 bytes per sector, a \$80 indicates 128 bytes per sector.  
 SDFS 2.1: additionally to SDFS 2.0 a 1 here indicates 512 bytes per sector. In this case everything else than \$80 is the high byte of the sector size measured in bytes minus 1.
- \$20 (32) File system revision number of the disk format. SD 1.1 has a \$11, SD 2.x, 3.x, SDX 4.1x and 4.2x all have a \$20 here, SDX 4.4x has \$21 here, indicating SDFS version 2.1.
- \$21 (33) SDFS 1.1: number of buffers reserved for sector storage.  
 SDFS 2.0: reserved - no known usage.  
 SDFS 2.1: sector size - LOW byte
- \$22 (34) SDFS 1.1: default drive used by the command processor if this disk is booted.  
 SDFS 2.0: reserved - no known usage.  
 SDFS 2.1: sector size - HIGH byte.
- \$23 (35) SDFS 1.1: reserved - no known usage.  
 SDFS 2.0: reserved - no known usage.  
 SDFS 2.1: number of sector entries per file map sector in low/high format. 2 bytes.
- \$25 (37) SDFS 1.1: this is the number of sectors in the main DOS boot  
 SDFS 2.0: reserved - no known usage.  
 SDFS 2.1: number of physical sectors per logical sector (cluster). Note that only one value - \$01 - is supported at the moment.
- \$26 (38) Volume sequence number. This number is incremented every time a file is opened for write on the disk. This is used to identify the disk.
- \$27 (39) Volume random number. This is a random number created when the disk is formatted. It is used with volume name and volume sequence number to positively identify a disk, to determine whether or not the data in the disk buffers is still valid.
- \$28 (40) Sector number of the first sector map of the file to be loaded when the disk is booted. This is usually a DOS-file. It is set by XINIT.COM and the BOOT command. 2 bytes
- \$2A (42) SDFS 2.0: This is the write LOCK flag. A value of \$FF indicates the medium is locked and a \$00 indicates that it is not.  
 SDFS 2.1: Not used (kept for backwards compatibility).

Values \$1F to \$25 (31 to 37) are recommended not to be changed. Bytes \$2A to \$2F (42 to 47) are reserved by SDFS 2.0 and should not be altered. For SDFS 2.1 bytes \$2A to \$3F (42 to 63) are reserved when using the 512 bytes per sector format.

**Note:** All DOS using SDFS 2.0 work with SDFS 2.1 and vice versa.

### 7.1.2 Bit Maps

A bit map is used to determine the allocation status of each sector on the disk. Each bit in every byte in the bit map shows whether the corresponding sector is in use. So each byte represents the status of eight sectors. Bit 7 represents the first sector of each group and bit 0 represents the eighth sector of each group. The bytes are in sequential order. Byte 0 of the first bit map sector represents sectors 0 through 7 (although sector 0 does not exist), byte 1 represents 8 through 15 and so on. If the bit representing a sector is SET (1), the sector is not in use. If it is CLEAR (0), then the sector is allocated. If more than one bit map sector is needed any additional bit map will follow on consecutive sectors.

### 7.1.3 Sector Maps

Sector maps are lists of the sectors that make up a file. The first two entries are the sector numbers of the next and previous sector maps of the file. The rest of the sector is a list of the sector numbers of the data sectors of the file or directory. The following are listed as offsets into the sector map:

- 0 The sector number of the next sector map of the file or directory. This will be 0 if this is the last sector map (2 bytes).
- 2 The sector number of the previous sector map of the file or directory. This will be 0 if this is the first sector map (2 bytes).
- 4 The sector numbers of the data sectors for the file in the proper order. If the sector number is 0, then that portion of the file has not been allocated. All sector numbers are two bytes long. It is also legal to sparse files<sup>81</sup> using the POINT command in e. g. BASIC.

## 7.2 Directory Structure

---

The directory structure of SDFS 2.1 is identical to version 2.0 but holds additional information. The first entry (the header) of the main directory has a valid date/time stamp: it is the date and time, when the file system has been built on that medium. The directory is a special file that contains information about a group of files and subdirectories. Each directory entry is 23 bytes in length and contains the file name, time/date, length, the number of the first sector map and the entry status. The first entry is different from the others; it contains information about the directory itself. The following is a list of this information given as offsets into the first entry:

- 1 The sector number of the first sector map of the parent directory. A 0 indicated that this is the main (or root) directory of the disk (2 bytes).
- 3 The length of the directory (in bytes). This is the length of the directory file, not the number of entries (3 bytes).
- 6 The name of the directory padded with spaces (8 bytes).

---

<sup>81</sup> Chapter 6 - Set file position -POINT.



When a directory is opened in unformatted or raw mode<sup>82</sup> the file is positioned to the second entry (that of the first file or subdirectory). To read the first entry you must POINT to the beginning of the file after opening it.

The rest of the directory entries are the same. They are 23 bytes long and provide the following information (given as offsets into the entry):

- 0 Status byte. The bits of this byte, if SET (1), represent the status of the directory entry as follows:
  - B0 - Entry is protected.
  - B1 - Entry is hidden.
  - B2 - Entry is archived.
  - B3 - Entry is in use.
  - B4 - Entry is deleted.
  - B5 - Entry is a subdirectory.
  - B7 - Entry is open for write.

**Notes:** bits 1 and 2 are not supported by earlier versions of SD. Bits 3 and 4 should always be opposites. Bit 5 should never be changed! A status byte of 0 indicates the end of the directory. Bits 6 is not used and should not be, since it may be cleared as other operations are performed.

- 1 The sector number of the first sector map of the file or subdirectory (2 bytes).
- 3 The length of the file in bytes (3 bytes).
- 6 The name of the file or subdirectory, padded with spaces if necessary (8 bytes).
- 14 The extension of the file or subdirectory, padded with spaces if necessary (3 bytes).
- 17 The date the file or directory was created in DD/MM/YY format (3 bytes).
- 20 The time the file or directory was created in HH/MM/SS 24 hour military format (3 bytes).

### 7.2.1 Exploring Disks

The best way to become familiar with the SDFS is to use is SDX own sector editor 'Eddy'<sup>83</sup> and a test medium to explore.

A good combination of understanding the SDFS media structure and Eddy can prove to be invaluable for recovering files from disks with bad sectors or damaged directories.

### 7.2.2 PERCOM Extensions

SDX 4.4x recognizes an extension to the PERCOM standard. In the 5th byte of the PERCOM block (PERCOM+5) the previously unused 3rd bit (i. e. +\$08) has now meaning as follows: when set (1), it means, that the disk does not have sides nor heads, thus the 4th byte of the PERCOM block (PERCOM+4) does not carry the number of them, but instead it contains the third byte of the number of sectors per track. When

<sup>82</sup> Chapter 6 - Accessing the raw directory.

<sup>83</sup> Chapter 9 – Disk Tools.

this bit is 0, the value of that byte should be ignored for hard disks, i. e. when the number of tracks is 1 (some hard drives tend to return the number of their physical heads here).

### 7.3 Direct Disk Access

---

Some programs need direct access to the sectors on a disk bypassing the DOS - e. g. sector copiers. Since the DD 512 density was introduced, the first sector size is not always 128 bytes long. The usage of READ PERCOM is strongly recommended to determine the current disk configuration, as contrary to judging the size of the first sector.

The program below is an example of a subroutine that returns the information about the size of the sector number 1 in AX (low/high) for the drive number (\$01-\$0F) specified in accumulator:

```

ddevic = $0300          sta daux1
dunit  = $0301          sta daux2
dcmdn  = $0302          jsr jsoint
dstats = $0303          bpl success
dbufa  = $0304          ;error 139 means that the drive
dtimlo = $0306          ;does not know READ PERCOM cmd
dbyt   = $0308          ;so it can only do 128 BPS
daux1  = $030a          ;(it is an Atari 810 or 1050)
daux2  = $030b          cpy #139
jsoint = $e459          beq a810
buffer = $0400          ;cassette buffer          ;any other error is just an error
                                          cpy #$00
getbootsize
    sta dunit          ;unmodified 810 or 1050,
    lda #$31          ;128 bytes per sector
    sta ddevic        ;a810
    lda #'N          ;READ          lda #$80
PERCOM
    sta dcmdn          ldx #$00
    lda #$40          ldy #$01
    sta dstats        ;success          rts
    lda #<buffer      ;low byte of the sector size
    sta dbufa        lda buffer+7
    lda #>buffer      ;high byte of the sector size
    sta dbufa+1      ldx buffer+6
    lda #$07          ;if the BPS < 512, return 128
    sta dtimlo        cpx #$02
;amount of data: 12 bytes          bcc a810
    lda #$0c          ;or the returned value otherwise
    sta dbyt          ldy #$01
    lda #$00          rts
    sta dbyt+1
;this should be zeroed because of
;some floppy turbo systems (e. g.
;Top Drive, TOMS Turbo)

```

## 8 Configuring Your System

This chapter contains all the information needed to configure your system the way you want it. There are many features and a lot of drivers for various functions that can be installed into the system. Of course, if you install all of these, you may not have sufficient memory left to program with or load a particular application. Therefore it is recommended to have at least 128 KiB of RAM in your A8 machine because it allows to reserve some extended memory for DOS usage.

### 8.1 SDX Boot Process

During the boot process SDX does quite some checks invisible to the user. Among those it will look up the drive number to boot from and the memory configuration, both may be customized. Additionally, it uses some defaults if the user does not provide own configurations to setup SDX during boot.

#### 8.1.1 Boot Drive Number

SDX forces the boot drive number to D1: only when it was not yet determined upon entering the DOS initialization routines. Normally, the XL OS does not select the boot drive number at this stage of system start-up, but it can be determined by the BIOS of a PBI hard drive. In such a case, the DOS will boot from the preselected disk rather than from the D1:.

#### 8.1.2 Memory Configuration

If not customized, SDX will handle available memory by the USE command with these default settings:

```
USE NONE = 400/800 48 KiB
USE OSRAM = XL/XE 64/128 KiB
USE BANKED = all Axlon machines, XL/XE with more than 128 KiB
```

They apply if no CONFIG.SYS file is found during the boot process. Nevertheless, to keep SDX working without the user's guidance it comes with a default CONFIG.SYS which lacks the USE command on purpose:

```
DEVICE SPARTA OSRAM
DEVICE SIO
DEVICE ATARIDOS
DEVICE INDUS 4
DEVICE RTIME8 (depends on the hardware version of SDX)
DEVICE JIFFY
DEVICE RAMDISK
```

SDX then applies the machine relevant default USE command before executing the default CONFIG.SYS file.

### 8.2 CONFIG.SYS File

SDX carries a default text file of configuration information called CONFIG.SYS on the CAR: device<sup>84</sup>. To override the use of the default configuration settings create a file

---

<sup>84</sup> Type TYPE CAR:CONFIG.SYS to see the default configuration file of your SDX.

called 'CONFIG.SYS' and save it as a text file in the 'MAIN' directory on the boot drive, which must be a SDFS formatted medium. The ED command comes in handy now.<sup>85</sup>

The 'CONFIG.SYS' file basically consists of commands. The current five commands are

```
USE OSRAM | BANKED | NONE
SET var=env_string
DEVICE driver
ECHO string
MERGE [d:][path]fname[.ext]
```

### 8.2.1 USE Command

The USE command should be the first command in your 'CONFIG.SYS' file as it instructs what secondary RAM area to use. OSRAM refers to RAM under the OS, BANKED refers to the banks of available extended memory. NONE refers to low memory, above the normal low memory part of SDX and below program memory. USE NONE will probably be incompatible with many programs, since it uses up a great deal of main memory. Any line beginning with a semicolon (;) is considered a comment and therefore ignored.

**Notes:** The size of the OSRAM memory area is 7 KiB (\$E400-\$FFBF) and the BANKED memory area is 16 KiB (\$4000-\$7FFF). If you have BANKED RAM, it is generally best to 'USE BANKED' unless you have a stock 130XE and wish to use BASIC XE in EXTEND mode (or any other programs that require the extra 64 KiB of RAM). If you do choose OSRAM, you may use the 4 KiB of RAM from \$C000-\$CFFF as buffers for the SPARTA.SYS driver (see its driver description).

Generally, the low free memory pointer (MEMLO) should never go higher than \$2000 for most programs to be executed. When its value is higher, the 'Memory Conflict' error may occur much more often. The MEMLO value should be taken into account when installing fancy drivers, especially if the DOS is configured to run under the OS ROM (USE OSRAM) and the buffers are located in the main memory.

If the computer has no extended memory, or the RAM extension is to be used in a different way, the best solution is to put DOS buffers under the OS ROM (USE OSRAM / DEVICE SPARTA OSRAM in CONFIG.SYS). In such a case the MEMLO value remains relatively low (around \$1100 with SPARTA.SYS and SIO.SYS /C), so you can load more drivers.

Whilst estimating the MEMLO value you should take into account the fact, that this pointer is raised by the X.COM program, which in turn is necessary to run most application programs. Therefore it is a good practice to do the command LOAD X.COM first and then check the MEMLO state (with MEM command).

If you put USE OSRAM in your CONFIG.SYS, regardless of parameters passed to the SPARTA.SYS, the file structures will always be located under the OS ROM and specifically under the Math Pack ROM (\$D800-\$DFFF). This is done in order to release some main memory (16 file structures take 640 bytes).

If a tailored CONFIG.SYS in CAR: device is required, please see the SDX Imager.<sup>86</sup>

<sup>85</sup> Chapter 4 – ED command.

<sup>86</sup> Chapter 10 – SDX Imager.

### 8.2.2 SET Command

The SET command is identical to the SET command in the command processor. You may set environment variables such as \$CAR, \$BASIC, or \$BATCH to default values. The environment variables \$CAR, \$BASIC and \$TEMP are defined by the RAMDISK.SYS driver, but only if the user did not define them before.

The \$COMSPEC variable is not defined, but its meaning and function remain the same as in earlier versions of the DOS. It contains the pathname of the shell.

### 8.2.3 DEVICE Command

The DEVICE command will load installable drivers such as SPARTA.SYS, RTIME8.SYS, etc. Each of these drivers is documented in this chapter later on.. Note that order is important (e. g. SPARTA.SYS must load before ATARIDOS.SYS).

If you hold down <OPTION> when booting the computer, any CONFIG.SYS on media will be ignored and the default configuration will be used. This is very useful if you happen to forget to include SIO.SYS in your CONFIG.SYS or some similar fatal error.

### 8.2.4 ECHO Command

The keyword ECHO enables the display of a simple message during the execution of the CONFIG.SYS.

### 8.2.5 MERGE Command

Generally, it allows merging of text files with CONFIG.SYS from any legal drive and path in the system. Therefore it is possible to have a modular built CONFIG.SYS. It works in two ways: basic use with OS routines or advanced use with SDX device drivers. Its use is optional, but if used, MERGE must be the last keyword in the current configuration file, because it aborts it and merges another file. That allows to form a chain of files to be processed at system startup with virtually no limits, except each one cannot exceed 1 KiB in size. Holding <OPTION> during system startup disables merging.

#### 8.2.5.1 Basic usage

Routines from the OS are used to read in the first file from an OS-compliant boot drive. Its name has to be CONFIG.SYS and it has to be placed in the root directory. If found there, the default CONFIG.SYS file in CAR: device will be skipped and CONFIG.SYS from the boot drive will be loaded and processed. Example:

```
USE BANKED
MERGE DEFAULTS
```

The system will configure the memory and then load a file named DEFAULTS.CFG from the root directory of the boot drive. That file should contain actual configuration commands - and also may contain another MERGE at the end, if it is necessary to merge a configuration part from yet another file.

The rules for the use of MERGE as shown in the above example are that the merged file must be located in the same directory as the file, that merges. It also cannot be used before the USE keyword occurs in the stream of configuration commands, if applicable. The file name extension (\*.CFG) is optional, but only 'CFG' is recognized automatically. Additional use of the config selector enhances the possibilities of the basic usage. See respective section on the following page for more details.

### 8.2.5.2 Advanced use

Again the first config file is read using OS routines, so above rules apply. But now SDX drivers may be used to get the next part of a config file residing in a non-standard location. Example:

```
USE BANKED
DEVICE SPARTA
DEVICE SIO
MERGE D3:\CONFIGS\DEFAULTS
```

The system will configure the memory, load the driver 'SIO.SYS', and then load a file named DEFAULTS.CFG from the subdirectory 'CONFIGS' on drive 'D3:'. That file should contain valid configuration commands - and may contain another MERGE at the end, if another configuration part from another file needs to be merged. The following parts to merge may reside on any via SIO.SYS driver available drive. Other devices can be accessed as soon as the respective driver is loaded (also MyIDE, SIDE, etc.). The advantages of the config selector apply here as well. But SDX still cannot boot from non-OS-compliant drives.

### 8.2.5.3 Special usage

When using non-OS-compliant 'boot' devices SDX cannot detect the drive nor find the CONFIG.SYS in the main directory, or make use of the config selector. It simply cannot boot from devices like SIDE, MyIDE, etc.

To overcome these obstacles a customized CONFIG.SYS in CAR: device is needed. It could read like:

```
DEVICE SPARTA OSRAM
DEVICE SIDE
MERGE
```

If extended memory is available it may be used, of course, by putting USE BANKED as the first line in and deleting the OSRAM parameter.

Since a customized default CONFIG.SYS in CAR: device now controls the boot process, a CONFIG.SYS in the main directory is of no use. Here again the config selector comes in handy. Notice the MERGE command without any filename in the above example. MERGE used this way invokes SDX to look for a directory named SPARTA.DOS on the drive now being mounted, and, if found, puts the config selector to use. Alternatively, you may wave these advantages and specify the name of a file to be merged from any legal drive and path.

To customize the contents of the CAR: device the SDX Imager<sup>87</sup> comes in handy. If the SDX driver for your special device is not found on CAR: you can put it there.

MERGE is a very useful feature in conjunction with multiple configuration files managed by the config selector. Such configuration files, when they differ only at the beginning, may define differences in a small number of commands. Next, merge the rest of the contents from common source. This allows to keep several configurations consistent by editing only a single text file.

### 8.3 Config Selector

SDX offers a built-in configuration selector. This feature allows the user to store several alternative CONFIG.SYS files on media and decide at boot time, which one is to be used instead of the default one. Of course this feature only works on SD formatted media.

At boot time the main directory of the boot medium is searched for a subdirectory named SPARTA.DOS. If found and containing \*.CFG files, a menu is displayed, where each \*.CFG file (up to nineteen) is assigned a letter. Hitting the appropriate letter key highlights the chosen file to be used to configure the DOS. The choice needs to be confirmed by pressing <RETURN>, otherwise the process is paused.

Waiting a few seconds without pressing any key causes the DOS to close the menu and proceed automatically depending on the capabilities of the system it runs on.

On system not able to store the last CFG file used the standard procedures using the default CONFIG.SYS file will be executed by SDX.

On systems able to store the last selection - currently supported by KMK/JŽ IDE 2.0 Plus and Ultimate 1MB - SDX will present it on the next boot as preselected choice being highlighted and waiting for the confirmation by <RETURN>. If no key is pressed now, the preselected configuration file will be used as default.

Pressing <ESC> causes SDX always to execute the default CONFIG.SYS.

**Note:** If the directory SPARTA.DOS is detected, a CONFIG.SYS file in the main directory of the boot drive is ignored.

### 8.4 Character Sets

When SDX is configured to use the RAM under the OS ROM for buffers (USE OSRAM / DEVICE SPARTA OSRAM in CONFIG.SYS), a problem may occur with the international character set (CHARSET 2). A copy of the character set is made in the RAM shadowing the OS ROM to avoid ugly screen effects when the memory is being banked. However, the same memory area is allocated for DOS buffers. When the number of buffers exceeds a certain limit (i. e. when they are more than 6), the font gets overwritten and the previously mentioned screen effects will occur.

The solution is to keep the number of buffers equal or lower than 6 (the default is 4) in this configuration. Again, these remarks apply, when CHARSET2 is in use **and** DOS buffers are under ROM.

### 8.5 Important system variables

Some global system settings are controlled with environment variables (type SET at the DOS prompt to see a list). In principle, the user can define own variables giving them arbitrary names. In reality, however, a part of the names is reserved. Values of these reserved variables control some parameters of the DOS and programs residing on CAR:. The list of variables reserved for SDX 4.4x is:

**Note:** Apart from those listed hereafter, all variable names beginning with '\_', ':', and '%' are reserved too.

**\$BASIC**

Defined by the user or RAMDISK.SYS  
 Controls CAR.COM  
 Remarks Contains the full path specification of the file, where the memory of the internal Atari BASIC module will be saved, when the 'DOS' command is executed. Page 0, 4, 5, 6 and everything from LOMEM to APPMHI will be saved. On return to BASIC the memory will be reloaded from that file.

**\$BATCH**

Defined by the kernel  
 Controls COMMAND.COM  
 Remarks Contains the name of the batch file, that will be executed after the COMMAND.COM is loaded for the first time. The default value is 'AUTOEXEC'.

**\$BOOT**

Defined by the kernel  
 Controls user programs  
 Remarks Contains the path to the main directory of the boot disk. On most configurations its value will be 'A:>'. Some programs, which e. g. install something on the hard disk, may be interested in this information. There is no need to change the value of the variable.

**\$CAR**

Defined by the user or RAMDISK.SYS  
 Controls CAR.COM  
 Remarks Contains the full path specification of the file, where the memory of an external cartridge (e. g. Action!) will be saved, when the user returns to the DOS. Page 0, 4, 5, 6 and everything from MEMLO to MEMTOP will be saved. Returning to the cartridge will cause the memory to be reloaded from that file.

**\$COMSPEC**

Defined by the user  
 Controls the I/O library  
 Remarks Points to the binary file, which will be the DOS shell. It will be loaded instead of the COMMAND.COM. Such a program must meet some conditions: first, it should be a relocatable SD module; second, it must communicate with the I/O library in the same manner as COMMAND.COM does, i. e. react appropriately on the FLAG register states and reply with appropriate status codes returned to the U\_FAIL routine. And it should perform basic shell tasks like offering file management functions and program execution.

As of SDX 4.42 the MENU can serve in a limited way as a replacement to the COMMAND.COM

**\$COPY**

Defined by the user  
 Controls COMMAND.COM  
 Remarks Points to the binary, which will be executed instead of the default COPY command. See COPY Command for details.



**\$DAYTIME**

Defined by the user  
 Controls SPARTA.SYS, COMMAND.COM, TD.COM  
 Remarks Controls the date and time format. There are three values allowed: '0' - the default format, '1' - the US format (MM-DD-YY and 12-hour clock); '2' - the EU format (DD-MM-YY and 24-hour clock). Default format '2'.

**\$DIRCOLORS**

Defined by the user  
 Controls COMMAND.COM  
 Remarks Defines colors for directory listing: 'a,b,c,d' for 'directory, file, protected directory, protected file' respectively. Of course it only works when the display device is able to display the colors and the software driver is loaded, which understands the requests to change colors generated by the program. 'SET DIRCOLORS=a,b,c,d' to get the color scheme as desired.

**\$ED**

Defined by the user  
 Controls ED.COM  
 Remarks Contains the numeric value, that defined the default number of lines visible in the editor window. See the ED command description for details.

**\$MANPATH**

Defined by the user  
 Controls MAN.COM  
 Remarks Contains the list of directories (separated by commas or semicolons) where the MAN command searches for documentation files. See the MAN command description for details.

**\$PAGER**

Defined by the user  
 Controls MAN.COM  
 Remarks Contains the command line template, which MAN.COM uses to execute external text viewer (such as LESS). See the MAN command description for details.

**\$PATH**

Defined by the kernel  
 Controls SPARTA.SYS, I/O library  
 Remarks Contains the list of directories - separated with commas or semicolons - where the system has to search for executables. See the PATH command description for details.

**\$PROMPT**

Defined by the kernel  
 Controls COMMAND.COM  
 Remarks Contains the prompt template for the Command Processor. See the PROMPT command description for details.

**\$RAMDISK**

Defined by RAMDISK.SYS  
 Controls user programs  
 Remarks Points to the main directory of the first ramdisk installed by RAMDISK.SYS. The default is 'O:>'

**\$SCRDEF**

Defined by the user  
 Controls COMMAND.COM  
 Remarks Defines default colors for foreground, background, frame, left and right margin. The command processor will set these values only when the variable is defined. Setting the variable itself by

```
SET SCRDEF=[COLPF1],[COLPF2],[COLBK],[LMARGN],[RMARGN]
```

When LMARGN is smaller than RMARGN, or the difference between the two values is less than 32, new margin settings are ignored.

Any number can be omitted, e. g. how to set margins to 0 and 38 not touching the colors: SET SCRDEF=,,,0,38

Setting SCRDEF alone does nothing. For it to take effect reload the COMMAND.COM e. g. with EXIT.

**\$SYSERR**

Defined by the kernel  
 Controls I/O library  
 Remarks Points to the text file, from where the system error messages are fetched. The default value is 'CAR:SYSERR.MSG'. The user can delete the variable - in this case the messages will only contain the error number and the text 'System error' - or change the value in order to replace the default file.

The system error messages are stored in a plain-text file. Its consecutive lines contain messages for errors, starting from error 128 in line number 0. An empty line (or rather: shorter than 5 characters) means no message: the default 'System error' will appear in this case. The same generic message will appear for any error code greater than the last one defined in the file.

It is recommended to put the SYSERR.MSG replacement on a really fast storage media, e. g. a ramdisk or hard drive.

**\$TEMP**

Defined by the user or RAMDISK.SYS  
 Controls COMMAND.COM, user programs  
 Remarks Points to the directory, where temporary files will be stored. It is recommended to locate this directory on a fast storage media.

**\$X**

Defined by the user  
 Controls X.COM  
 Remarks Reverts the operation of the switch '/C'.

## 8.6 Drivers

With SDX 4.4x a lot of all new features have been introduced to the A8 world. Therefore quite some efforts have been made to update drivers kept from version 4.2x and write new ones as deemed necessary for the upgraded system.

### 8.6.1 File System Drivers

Currently, the main file systems in the A8 world are supported, as there are the ones from SD and AtariDOS 2 and derivatives. For other file systems (e. g. AtariDOS 3) please use available converters to AtariDOS 2 first. Up to eight such drivers can be installed at a time<sup>88</sup>.

#### 8.6.1.1 SPARTA.SYS - SDFS disk format

---

##### Purpose

This is the SD format disk driver. It must be installed - if it is not, your system will have no purpose (i. e. no way to read/write to disks).

##### Syntax

```
DEVICE SPARTA [OSRAM] [nbufs [,nfiles]]
```

##### Type

External - on CAR: device.

##### Remarks

This is the largest of all the drivers and contains 3 subprograms, these are

- the SD 'kernel' functions,
- the formatted directory output and other miscellaneous functions (the MISC vector), and
- the default block I/O functions (the BLOCK\_IO vector).

The 'OSRAM' parameter only applies if the system is set to 'USE OSRAM'; it will be ignored otherwise. In this mode, the memory from \$C000-\$CFFF will be used for sector buffers, otherwise they will be allocated from main RAM. The default is to not use 'OSRAM'.

Because of the added support for 512-byte sectors the buffers (nbufs) have been enlarged to 512 bytes each. The maximum number of them has simultaneously been reduced to eight in some configurations (USE NONE and USE OSRAM with the buffers kept under the OS ROM). When the main or banked memory is allocated for buffers, the maximum is 16. One cannot declare less than 3 buffers, default number is 4.

It should be kept in mind, that 16 buffers, 512 bytes each, take twice as much memory as the same number of 256-byte buffers. If the DOS is told to USE BANKED, and 16 buffers are declared, it is quite likely that the extended memory bank the system uses will get completely filled up - in this case any drivers loaded afterwards will occupy the main memory, and the MEMLO will get raised. A good practice then is to never declare more than 12 buffers, unless a bigger number of them is really required.

---

<sup>88</sup> Chapter 9 - SDXTK, FS drivers.

The 'nfiles' parameter is the maximum number of disk files that may be open at one time, ranging from 2 to 16 - the default is 5. Each number here takes up 40 bytes of memory. If you USE BANKED this will be taken from the DOS bank. If you USE OSRAM and use the OSRAM parameter, this will be taken from \$C000-\$CFFF until that area is full and from low memory (raising MEMLO) after that. If you USE NONE (or USE OSRAM without using the OSRAM parameter for DEVICE SPARTA) this will be taken from low memory (RAISING MEMLO).

Unlike other DOSes, where a buffer is usually assigned in a fixed manner to an open file, the SDX buffering mechanism is a sort of a sector cache. This cache is maintained by the SPARTA.SYS driver to keep last accessed sectors, regardless of their type, i. e. whether these are boot sectors, bitmap sectors, file map sectors, data sectors or whatever. The greater the number of buffers, the less often the DOS is forced to re-read required data from the actual media. So, decreasing the number of buffers is unlikely to cause errors, but it certainly will make the file system work slower.

If you get an error 161, you need to increase the number of file buffers. This is done in the CONFIG.SYS file with the SPARTA.SYS driver as described. Just increase the 'nfiles' value by one or more. Increasing 'nbufs' will speed up medium access for additional open files but is not required.

SPARTA.SYS will check the physical sector size and return error 176 (Access denied) when anything bigger than 512 bytes is detected.

### **8.6.1.2 ATARIDOS.SYS - AtariDOS 2.0 file system**

---

#### **Purpose**

This driver contains the code to recognize AtariDOS 2 format media. This driver also supports the various derivatives of DOS 2 including MyDOS and DOS 2.5.

#### **Syntax**

DEVICE ATARIDOS

#### **Type**

External - on CAR: device.

#### **Remarks**

This driver requires that the 'SPARTA.SYS' driver has been previously loaded (it is like an extension to the 'SPARTA.SYS' driver). It supports all the derivatives of AtariDOS 2 including subdirectories in MYDOS up to a size of ~16 MiB (65535 sectors, 256 bytes each). It supports the extended sectors of DOS 2.5 for read only.

It does not support the ability to create/delete, or set a working directory on MYDOS media. ATARIDOS.SYS has no support for DOS 3, DOS XE, or OSS version 4 DOS.

For compatibility reasons ATARIDOS.SYS will now mark zero-length files in the directory as occupying 1 sector, not 0 sectors as before.

## 8.6.2 Block I/O Drivers

### 8.6.2.1 SIO.SYS - serial and parallel I/O

---

#### Purpose

This is the high speed SIO and parallel I/O driver. It is a required driver as there is no default SIO driver.

#### Syntax

```
DEVICE SIO [/C|A]
```

#### Type

External - on CAR: device.

#### Remarks

This driver must be included in the 'CONFIG.SYS' file since it contains all the code to handle high speed SIO operations with e. g. US Doubler, Indus GT, and all other high speed drives, if recognized. It also handles the standard speed SIO with all other drives and the standard parallel I/O (PIO) with devices such as the KMK/JŽ IDE 2.0 Plus, MIO, MSC, etc. DEVICE SPARTA must precede DEVICE SIO in CONFIG.SYS.

It has been greatly improved over the earlier versions. First of all, an Ultra Speed drive is asked first, what serial speed it prefers (the old SIO was fixed at 52 kbps). Next, once the US mode is enabled, the SIO does not fall back to 19,200 bps so easily, when an error occurs. So the TOMS drives can spread invalid responses around as they usually do and the transmission still remains at the turbo baud rate. If the automatically selected speed turns out to be invalid anyway, you can still change it manually using SIOSET<sup>89</sup>.

The SIO.SYS driver will no longer periodically re-query drives for high speed by default. If you want such behavior, issue SIOSET REFRESH 128 (0 is the default). However, the refresh may affect manual high speed setting (SIOSET NORMAL etc.).

Additionally, there is now a built-in mechanism for the 'mapping' of drives, accessible by MAP<sup>90</sup>.

If all that is waived to go for the maximum free memory, SIO.SYS can be loaded with the '/C' option (for 'classic'). This will load the old-fashioned SIO.SYS driver as known from SDX 4.20. The only change to it is the capability to handle 512-byte sector devices.

If even the classic SIO is not needed, the SIO routines residing in the Atari ROM can be used. Passing the '/A' option (as 'Atari') to the SIO.SYS driver will do. It will occupy a minimum of memory, but transmission parameters will depend on the OS capabilities.

**Notes:** Please remember that using the '/A' or '/C' option disables mapping as provided by the map command.

The serial I/O driver will always have the lowest possible priority among all the other SIO alike drivers (ramdisks, SIDE/MyIDE drivers, etc.), regardless of the order of loading.

---

<sup>89</sup> Chapter 4 - SIOSET.

<sup>90</sup> Chapter 4 - MAP.

### 8.6.2.2 INDUS.SYS - Indus high speed SIO

---

**Purpose**

This is the high speed SIO installer for Indus drives. It is required for high speed operation with Indus GT, CA-2001, LDW Super 2000, and Happy drives.

**Syntax**

DEVICE INDUS [n]

**Type**

External - on CAR: device.

**Remarks**

This driver takes up no memory, it simply scans for existing drives and programs them with the appropriate high speed code. Optionally, you may limit the number of drives (D1: - Dn:) to be scanned for. Once the found drives are programmed, they will stay programmed until power is shut off on the drives. Thus, the drives do not need to be programmed every time the computer is booted. Also the 'SIO.SYS' driver needs to be installed before the 'INDUS.SYS' driver.

This driver is required for Happy drives, because it sends them a command, that makes sure that the drive will work properly (it 'bug-fixes' the Happy firmware this way).

**Note:** In the default CONFIG.SYS the number of drives to scan for the INDUS.SYS has been limited to 4 (D1: - D4:). If you want it to act by default on higher drives, edit the CAR:CONFIG.SYS and change or remove the parameter digit in DEVICE INDUS line.

### 8.6.2.3 RAMDISK.SYS - SDX ramdisk

---

**Purpose**

The SDX ramdisk driver. Specifies drive number and size of up to three ramdisks .

**Syntax**

DEVICE RAMDISK [drive[,nbanks]] [/S]

**Type**

External - on CAR: device.

**Availability**

As of SDX 4.40 uses 65C816 block move instructions if the new CPU is present.

**Remarks**

RAMDISK.SYS defaults to the O: drive. Different drive numbers and appropriate sizes need to be given to install multiple ram drives. An attempt to select more banks of RAM than are available will use all available RAM. To check the number of 16 KiB RAM banks in the current system, see the MEM command. Additionally, if the user did not define them otherwise, the RAMDISK.SYS driver defines environment variables like \$BASIC, \$CAR, \$RAMDISK and \$TEMP so that they point to appropriate filenames and their drive numbers.

If there is no custom CONFIG.SYS in the boot sequence, the standard setup from the CAR: device will install one ramdisk as drive O: and build the directory structure on it. The size by the standard setup is

- XL/XE: all available windowed RAM minus 4 banks (64 KiB) reserved for BASIC XE, and minus 1 bank used by SDX for DOS routines and drivers.
- 800: all available windowed RAM minus 1 bank used by SDX for DOS routines and drivers.

Restarting the system with the COLD command will not destroy the ramdisk nor its contents, since drive number and size will still be the same. This applies as well when using a custom CONFIG.SYS. During the restart a corresponding message for each ramdisk will be shown on the screen:

Ramdisk preserved

Any attempt to install more than three ramdisks either by a custom CONFIG.SYS or directly from the command line interface will cause the error

Ramdisk not installed!  
SIO table full!

The switch '/S' forces loading the standard 6502 RAM driver module on 65816 machines instead of the 65816 RAM driver. The 6502 module is much slower, but has an advantage of occupying much less memory than 65C816 mods.

If you have more than 64 KiB of extended memory in your XL/XE computer, SDX will by default automatically use one of the banks (USE BANKED) for DOS routines and drivers. This means that there will be one bank less for the ramdisk. A custom CONFIG.SYS specifying USE OSRAM will allow the use of all available banks for the ramdisk.

The default configuration is 'DEVICE RAMDISK', utilizing all available banks beyond the four reserved for 130XE programs and the one SDX uses (when there are more than 64 KiB of extended memory) and assigning the ramdisk to drive O:. This will also be invoked when 'nbanks' is erroneously set to 256.

You can change this in a custom CONFIG.SYS file by specifying the drive number and number of banks. To override the reservation of the four banks for XE programs, the number of banks must be specified in the 'DEVICE RAMDISK' statement.

When the number of banks specified is greater than 255, SDX will now assume the default number of banks to be allocated<sup>91</sup>.

**Notes:** Holding down <OPTION> when booting the computer causes the system to ignore any custom CONFIG.SYS, using the default configuration instead. This is very useful if you happen to forget to include SIO.SYS or some similar fatal error.

RAMDISK.SYS has no effect on Multi I/O (MIO) extended memory.

### 8.6.2.4 PBI.SYS - special parallel I/O driver

---

**Purpose**

Provide additional support for PBI devices.

**Syntax**

DEVICE PBI

**Type**

External - on CAR: device.

**Availability**

As of SDX 4.40.

**Remarks**

The PBI.SYS driver improves support for parallel bus devices, such as hard disks. Some programs may fail to load from such a device because they require the data to be loaded to the memory, that shadows the PBI ROM area. The parallel device driver residing in that ROM naturally is not able to write data under itself. The PBI.SYS solves this (rare) problem.

**CAUTION:** MAP and SIOSET commands do not handle PBI bus devices with a driver installed.

### 8.6.3 Timekeeping Drivers

Without any clock driver installed, the TIME/DATE commands show the time and date of the last file loaded. Mostly this will be the SDX revision time and date.

#### 8.6.3.1 ARCLOCKS.SYS - Atari Real Clock

---

**Purpose**

Driver for the ARC realtime clock.

**Syntax**

DEVICE ARCLOCKS [/F]

**Type**

External - on CAR: device.

**Availability**

As of SDX 4.40.

**Remarks**

The driver will check for the battery packed Atari Real Clock (ARC). This handler will not load, when the hardware is not recognized (i. e., it is not plugged in) or another installed clock driver is detected.

The '/F' switch lets it bypass these tests and any already installed clock driver will be overwritten.



The driver will reset the ARC to '1-Jan-2000 00:00:00', when it discovers that the current setting is invalid.

**Note:** If the driver does not recognize an attached and working ARC and responds 'ARC clock not present', please try the /F switch.

### 8.6.3.2 IDEPTIME.SYS - IDE 2.0+ RTC

---

#### **Purpose**

Driver for the realtime clock residing on the KMK/JZ IDE V 2.0 Plus interface.

#### **Syntax**

DEVICE IDEPTIME [/F]

#### **Type**

External - on CAR: device.

#### **Availability**

As of SDX 4.44.

#### **Remarks**

The driver will check for the real time clock on the KMK/JZ IDE 2.0 Plus IDE interface. This handler will not load, when the hardware is not recognized (i. e., it is not plugged in) or another installed clock driver is detected.

The '/F' switch lets it bypass these tests and any already installed clock driver will be overwritten.

### 8.6.3.3 ULTIME.SYS – Ultimate, SIDE 1&2, SuperCart RTC

---

#### **Purpose**

Driver for the realtime clock residing in the respective cartridges.

#### **Syntax**

DEVICE ULTIME [/F]

#### **Type**

External - on CAR: device.

#### **Availability**

As of SDX 4.44.

#### **Remarks**

The driver will check for the real time clock in the Ultimate, SIDE, SIDE2 and SDX SuperCart. This handler will not load, when the hardware is not recognized (i. e., it is not plugged in) or another installed clock driver is detected.

The '/F' switch lets it bypass these tests and any already installed clock driver will be overwritten.

### 8.6.3.4 JIFFY.SYS - Atari software clock

---

**Purpose**

Driver for the Jiffy software clock in SDX.

**Syntax**

DEVICE JIFFY

**Type**

External - on CAR: device.

**Remarks**

If there is already another clock handler installed this handler will not be installed.

Use this clock driver if there is no realtime clock in your system.

During the boot process time and date should be set appropriate otherwise the system's time and date will be derived from the last file loaded. Of course one can do this also manually from the command line.

### 8.6.3.5 RTIME8.SYS - R-Time 8 RTC

---

**Purpose**

Driver for the R-Time 8 realtime clock from ICD.

**Syntax**

DEVICE RTIME8

**Type**

External - on CAR: device.

**Remarks**

If there is already a clock device handler installed or if no R-Time 8 cartridge is plugged into the cartridge port, this handler will not load.

**Notes:** The driver may be used with the emulator 'Altirra'. RTIME8.SYS does not load when Maxflash8 hardware is used. This has been introduced to prevent hang-up caused by hardware conflicts.

### 8.6.3.6 Z.SYS - access to SDX timekeeping functions

---

**Purpose**

Provide a Z: device compatible with SD 3.2.

**Syntax**

DEVICE Z [/I[S]

**Type**

External - on CAR: device.

**Availability**

As of SDX 4.40.

**Remarks**

A 'Z:' device is created in the OS handler table to offer a simple interface to SD timekeeping functions, accessible e. g. from a BASIC program. The device is compatible with the driver for SD 3.2 (ZHAND.COM). Therefore software using it should have no problems accessing the desired functions under SDX anymore.

Z.SYS features four internal functions selected with BASIC XIO instructions (or appropriate OS directives):

- 1) XIO 33: read time, unformatted
- 2) XIO 35: read date, unformatted
- 3) XIO 36: set time
- 4) XIO 37: set date

```
Read time:  10 OPEN #1,4,0,"Z":REM open for read
            15 REM select reading time
            20 XIO 33,#1,4,0,"Z:"
            25 REM get the clock state
            30 GET #1,H:GET #1,M:GET #1,S
            35 CLOSE #1
```

```
Setting time: 10 OPEN #1,8,0,"Z":REM open for write
              15 REM select setting time
              20 XIO 36,#1,8,0,"Z:"
              25 REM set the clock
              30 PUT #1,H:PUT #1,M:PUT #1,S
              35 CLOSE #1
```

The procedure to get and set the date is identical, you just have to set the XIO function codes to 35 and 37 respectively. An attempt to read or write more than 3 bytes causes the error 136 (EOF) to occur. To reset the read/write pointer you should close and re-open the device, or call the appropriate XIO function again.

The functions setting the clock are disabled by default, attempts to write to the 'Z:' device will cause the error 139 (NAK) to occur. Installing the driver with the /I switch (as in ignore) changes the returned status to \$01 (success), but nothing else is done. To enable these functions you should load the driver with /S switch (as in set) given as a parameter.

Z.SYS can only handle one I/O stream - an attempt to open more of them simultaneously will return error number 161 (Too many channels open).

Loading TD.COM enables more functions:

- 5) XIO 38: TD display line enable (TD ON)
- 6) XIO 39: TD display line disable (TD OFF)
- 7) XIO 34: read date, formatted
- 8) XIO 32: read time, formatted

These functions will only work, when TD.COM was loaded - or error 139 (NAK) will be returned otherwise. Example:

```

10 DIM TIME$(13):REM reserve at least 13 bytes
15 OPEN#1,4,0,"Z":REM open for read
20 REM read formatted time
25 XIO 32,#1,4,0,"Z:"
30 INPUT #1;TIME$
35 PRINT TIME$
40 CLOSE #1

```

Z.SYS requires a compatible clock driver to be loaded as prerequisite.

## 8.6.4 Screen Drivers

### 8.6.4.1 XEP80.SYS -Atari XEP 80 column display

---

#### **Purpose**

80 column display using the Atari XEP80 adapter.

#### **Syntax**

```
DEVICE XEP80 [1|2] [/P|/N]
```

#### **Type**

External - on CAR: device.

#### **Availability**

As of SDX 4.40 works with PAL computers.

#### **Remarks**

The XEP80 can now be connected to either joystick port. The first parameter is the port number to be used (1 or 2, default is 2). After installation, anything printed to the E: or CON: devices will be printed to the 80 column monitor through the XEP80. The XEP80 requires a 80 column monitor of its own. The regular 40 column output of the computer is unaffected, so that graphics output can be produced simultaneously.

Additionally, you can force either display mode by adding switches to the XEP80.SYS command line: [/P] for PAL or [/N] for NTSC.

**Notes:** XEP80.SYS does not provide a driver for the printer port on the XEP80.

Please be aware that many programs access screen memory directly and bypass the E: device. Almost all word processors are this way. These programs will not work through the XEP80. Instead, these will sent their output to the regular computer display. MENU.COM and FORMAT are good examples of this. Other programs use a combination of the two, such as the terminal programs 850 Express! 3.0 or BobTerm 1.2x. The actual online part of those will work properly on the 80 column screen, but the menus will show up on the 40 column screen. For these reasons it is a good idea to keep the 40 column monitor attached and running if space permits. A 80 column monochrome monitor is highly recommended for use with the XEP80.

### 8.6.4.2 QUICKED.SYS - accelerated screen output

---

**Purpose**

Provides an accelerated screen output in Graphics 0.

**Syntax**

DEVICE QUICKED

**Type**

External - on CAR: device.

**Availability**

As of SDX 4.40.

**Remarks**

QUICKED.SYS is a software screen accelerator. It installs into the CON: (DOS) and E: (OS) devices, speeding up the GRAPHICS 0 console operation up to four times.

**Notes:** When using the ACTION! cartridge there are no characters displayed on the command line of the monitor. Even not visible they will be conducted if typed in correctly. Please deactivate this driver when using ACTION!, BASIC XE and BASIC XL. They comprise a screen accelerator of their own.

When using CON64.SYS or CON.SYS (SDXTK), please be aware of the right order in CONFIG.SYS, where DEVICE QUICKED must precede DEVICE CON64 and/or DEVICE CON to get both working properly. And it is recommended to call DEVICE RAMDISK thereafter, otherwise one bank of extended memory is wasted.

### 8.6.4.3 CON64.SYS - 64-column display

---

**Purpose**

Provide a 64 column display.

**Syntax**

DEVICE CON64

**Type**

External - on CAR: device.

**Availability**

As of SDX 4.41.

As of SDX 4.46 enhanced to run on Axlon type memory extensions.

**Remarks**

This driver requires a compatible memory extension, either Port B or Axlon type. It is an experimental driver, that installs into the CON: and E: devices, and emulates a 64x24 text console in software using the 320x192 graphic display. After installing it defaults to the standard 40x24 text mode. While in the Command Processor, you can enable the 64-column text mode by typing 'CON 64' at the DOS prompt. To disable it type 'CON 40'.

The 64-column console is not very useful for Command Processor operations. First, the screen is in fact in GRAPHICS 8, the 320x192 bitmap mode and occupies nearly 8 KiB of the main memory. This sets the MEMTOP value at \$8035. Few programs are actually happy with this. Second, not every SDX utility can cope with the screen configured so. For example, as of SDX 4.42 MENU.COM works properly but ED.COM still fails miserably. Some of these problems may of course be fixed in future releases of the DOS.

The driver may be quite happily used in BASIC and in your own programs. (→ Chapter 6, 'Using CON: drivers in own programs').

The 64-column screen console functions identically to the normal 40-column one, just the logical line is longer and consists now of 64 characters instead of 40. Since a logical line, however, still covers a maximum of three physical screen lines, this sums up to 192 characters.

The driver also installs into the S: device. Under the control of the CON64 driver, there is no traditional form of text window anymore in any display mode - the 64-column console driver is not able to setup or handle such a window. The text can be more or less freely mixed with graphics. The operation of the GRAPHICS 0, 8 and 24 modes under the control of the driver is as follows:

- 1) GRAPHICS 0: this is the 64x24 text mode. In this mode, the BASIC's POSITION keyword is effective on the text cursor only. You will probably be able to draw points or lines on it using the OS' PLOT and DRAWTO functions, but it is not recommended since both S:, the display driver and E:, the console driver share the same screen coordinates. Therefore it is quite likely that an attempt to draw anything via the former will result in position range errors reported by the latter.
- 2) GRAPHICS 24: this is the 320x192 bitmap mode. In this mode the BASIC's POSITION keyword is effective on the graphic cursor only. You will probably be able to print text on it using appropriate commands of the E: device, but it is not recommended for same reason as above.
- 3) GRAPHICS 8: this is the 320x192 bitmap mode with text window. In this mode, just as in the GRAPHICS 24, the BASIC's POSITION keyword is effective on the graphic cursor only. The text cursor position can be controlled through OS variables TXTCOL (\$0291) and TXTROW (\$0290), for the x and y coordinates respectively. In this mode you can safely both print text and draw graphics, because the screen driver maintains two separate sets of coordinates for the text and graphic cursors.

**Note:** The 'text window' is not limited to the bottom three lines of the screen, but covers the entire graphic display. This has some side effects, such as clearing the text screen also clears graphics and vice versa.

## 8.6.5 Application Drivers

### 8.6.5.1 COMEXE.SYS - cartridge management

---

**Purpose**

Enables automatic cartridge management when launching programs.

**Syntax**

DEVICE COMEXE

**Type**

External - on CAR: device.

**Availability**

As of SDX 4.40.

**Remarks**

COMEXE.SYS is a system extension that distinguishes between \*.COM and \*.EXE type binaries causing the DOS to load them in slightly different manner. The \*.COM files are considered external commands and simply searched for and loaded as before. Now the \*.EXE files are searched for and loaded too, but before that the SD I/O library module is automatically switched off releasing the cartridge area at \$A000-\$BFFF. In other words, if a binary has an \*.EXE extension it is a signal for the DOS, that it should be executed using X.COM - the system can now do it automatically. No more need to care about typing in the extension at the DOS prompt.

Entering the '+' sign at the beginning of a command causes bypassing the COMEXE.SYS while executing the command.

### 8.6.5.2 RUNEXT.SYS - file association

---

**Purpose**

This provides file association support.

**Syntax**

DEVICE RUNEXT [d:][path][filename.ext]

**Type**

External - on CAR: device.

**Availability**

As of SDX 4.40.

As of SDX 4.47 allows assigning of multiple file types.

**Remarks**

RUNEXT.SYS is an extension to the Command Processor, that allows to define associations between file types and application programs. For example, if the \*.ARC files are associated this way with the ARC.COM archiver, and the user types in an \*.ARC filename at the command prompt, the Command Processor can automatically execute the archiver and hand over the specified filename along with predefined arguments to it.

The optional argument to the RUNEXT.SYS is a pathname to its configuration file. When none is given the default CAR:RUNEXT.CFG will be used.

The config file consists of lines defining one association each and of comments (a comment has a semicolon or an asterisk at the beginning). The format of a line defining an association is:

```
[!^]EXT,PROGRAM [,PARAMETERS]
```

where:

!,^ - optional modifier-key flags - '!' for <SHIFT> and '^' for <CTRL>.

**EXT** - filename extension (file type) to be associated.

**PROGRAM** - file name (with optional path) of the executable we associate with the file type above.

**PARAMETERS** - optional arguments to be handed over to the program; if nothing is defined here, the only argument passed to the program will be the data file name; if the file name has to be given at certain point of the command passed, we mark this place with a percent (%) character.

Example 1: ARC,CAR:ARC.COM,L %

This is an association for ARC archives. Such files will be opened using CAR:ARC.COM. The first parameter handed over to it will be 'L', the second - the archive file name. As a result, typing in an archive name at the DOS prompt, for instance:

```
D1:ARCHIVE.ARC
```

causes the archive's contents to be listed on the screen.

```
Example 2: !^ARC,CAR:ARC.COM,X %
          ^ARC,CAR:ARC.COM,V %
          !ARC,CAR:ARC.COM,P %
          ARC,CAR:ARC.COM,L %
```

This will create a group of associations for ARC archives. All of them will open the files using the CAR:ARC.COM program, but each one with different parameters.

Assigning of multiple file types (extensions) to a single entry in the RUNEXT.CFG file is also legal. Instead of assigning four file types to one program using the common syntax like this:

```
CMC,D:>MUZ>CMCPLAY>CMCPLAY.COM
DMC,D:>MUZ>CMCPLAY>CMCPLAY.COM
CMR,D:>MUZ>CMCPLAY>CMCPLAY.COM
CM3,D:>MUZ>CMCPLAY>CMCPLAY.COM
```

It is now possible to assign them in a much shorter form:



Example 3: `CMC:DMC:CMR:CM3,D:>MUZ>CMCPLAY>CMCPLAY.COM`

It does exactly the same, just saving (in this case above) over 80 bytes of memory.

**Notes:** Testing of associations is done in order of occurrence in the configuration file.

Entering the '+' sign at the beginning of a command causes bypassing the RUNEXT.SYS while executing the command.

## 8.6.6 Other Drivers

### 8.6.6.1 DOSKEY.SYS - extended command line

---

#### Purpose

Command line extension.

#### Syntax

```
DEVICE DOSKEY [d:][path][fname.ext] [/X]
```

#### Type

External - on CAR: device.

#### Availability

As of SDX 4.42.

As of SDX 4.45 DOSKEY handles filename completion.

As of SDX 4.47 DOSKEY is Axlon-compatible.

#### Remarks

DOSKEY requires extended memory to provide its capabilities. A history buffer for commands, command concatenation, definition of aliases and filename completion are now available.

#### Core functionality

If loaded, DOSKEY.SYS allocates one bank of extended memory (16 KiB) to record commands the user types at the DOS prompt (BAT file commands are not recorded). The buffer can hold up to 244 such command lines. After reaching this limit the oldest entries will be replaced with the new ones. The length of a line is limited to 63 characters plus 1 fixed EOL at the end. Empty lines are not recorded. The commands can be invoked with the following key shortcuts:

- <CTRL><P> (as 'Previous') or up arrow: invoke the previous (or older) command.
- <CTRL><N> (as 'Next') or down arrow: invoke the next (or newer) command.

Invoking a command from the history buffer does not replace its entry in the buffer. A new entry is created instead, identically as for a command typed in manually from the keyboard.

The switch [/X] will enable a special shortcut command. Pressing just <RETURN> will display the directory of the current drive.

Another special shortcut is <TAB>. If pressed on an empty command line, this will display all internal commands, aliases and symbols available to the user, and thereafter all binaries found in \$PATH.

DOSKEY will now clear the screen when you use the <SHIFT><CLEAR> combination.

**Note:** The active history buffer limits the set of control keys to edit the command line. Apart from the above, these are: <F1-F4>, <right arrow>, <left arrow>, <BACKSPACE>, <CTRL><INSERT>, <CTRL><DELETE>, <SHIFT><DELETE> and <RETURN>.

**Command concatenation**

DOSKEY enables to type more than one command in one command line. You must use the '&'-character as separator, for example:

```
CD DATA & DEL *.* & CD .. & RD DATA
```

This will execute these four commands in order from left to right. It is not necessary to put spaces around the '&'.

**Aliases**

This functionality offers a possibility to define alternative names or aliases to DOS commands. They are defined in a text file, whose filespec should be given to DOSKEY.SYS as a parameter. The definition consist of the alias, the '=' sign, and the proper command, that will be executed, when the user types in the alias and an EOL character. For example:

```
MOVE=COPY /M
```

Now when the user types in 'MOVE' at the beginning of the command line, the DOSKEY will substitute 'COPY /M' for that before passing the entire command line on to the DOS. This way you can execute entire command lines typing just one letter.

The maximum number of aliases is 256. When there are more, the program produces a warning and the superfluous aliases will get ignored.

An alias line may not be longer than 127 characters and it has to be terminated by EOL. Aliases are loaded to the history buffer decreasing its size accordingly. The size of this buffer is around 15 KiB, at least 1 KiB (16 entries) must remain free for the history. Exceeding this limit will produce an 'Out of memory' warning and all aliases will be removed from the memory. CAR:ALIASES.INI is an example file that defines two aliases.

**Filename completion**

Pressing <TAB> while typing a command name will cause a search through commands (known by COMMAND.COM), aliases and \$PATH for matching command names and file names. Pressing <SHIFT><TAB> tells the program to search also the hidden files (which are normally ignored). For example:

```
A:FI<TAB>
```

Since the only matching file is FIND.COM, this will cause the DOSKEY to supply this file name (i. e. 'FIND.COM'). After that the user will be allowed to continue typing. When more names match, their names will be displayed as possible choices to narrow the search pattern.

The commands, aliases and \$PATH are searched through only when typing a command (i. e. the first component of the command line). The strings typed after a separator are treated as parameters, and therefore matching files are only searched in one directory (the one pointed to by the path typed so far, or, if nothing has been typed that looks like a path, the current one). For example:

```
A:DIR >BIN>N <TAB>
```

assuming that the only matching file in the >BIN> is NEOPLAY.COM, will display:

```
>BIN>N*.*:
  NEOPLAY.COM
```

and resolve the parameter to:

```
A:DIR >BIN>NEOPLAY.COM
```

The path name may of course contain a device name. The DOSKEY will then act accordingly and do a search on the specified device.

It is possible to disable searching commands, aliases and \$PATH for the command stage. To accomplish that, you should press <CTRL><TAB> instead of <TAB> - in such a case the 'parameter search' will be performed instead of the command search. Similarly, in the parameter stage of command line, pressing <CTRL><TAB> performs a command search instead of the default parameter search.

**Note:** DOSKEY may now be controlled externally; this is an interface for programs respectively shells, which want to use the history buffer, aliases etc., but do not want to be blocked by DOSKEY waiting for user's input.

### 8.6.6.2 ENV.SYS - environment extension

---

#### **Purpose**

Environment extension

#### **Syntax**

DEVICE ENV

#### **Type**

External - on CAR: device.

#### **Availability**

As of SDX 4.42.

#### **Remarks**

Normally the area where the environment variables are kept is only 256 bytes in size. This can turn out to be too small for some applications. The ENV.SYS driver allocates one entire bank of the extended memory (16 KiB) for the environment. Thanks to that up to 128 variables, 128 characters each can be defined. We strongly recommend installing the ENV.SYS, if you plan to use pipes and batch files intensively.

**Notes:** For best results load ENV.SYS before SPARTA.SYS, when the DOS is configured to USE BANKED, or after SPARTA.SYS otherwise.

If used, ENV.SYS and SPARTA.SYS require one bank each. Therefore reserve 2 banks of extended memory in your memory configuration for them.

## 9 The SpartaDOS X Toolkit

Traditionally, the toolkit is a complement for the DOS, providing additional programs. These tools offer much more than the legacy versions of SDTK. It comprises:

- A set of more than 40 additional programs, drivers and utilities.
- A set of SDX command and driver descriptions for MAN utility, the man pages.

As of SDX V. 4.48 the main directory of the atr image (as of 27 June 2016):

```
Volume:   Toolkit
Directory: MAIN

BAT              <DIR>   6-09-10  11:22:18
BOOTLD          <DIR>   27-04-11  22:53:14
COMPAT          <DIR>   20-05-15  19:42:56
DLL             <DIR>   6-07-14  21:20:37
DRIVERS         <DIR>   27-04-11  22:49:55
DSKTOOLS        <DIR>   10-02-13   3:26:49
SDX42           <DIR>   20-05-15  19:42:13
SHELLS          <DIR>   10-02-13   3:25:10
UTILS           <DIR>   27-04-11  22:50:05
FILES   TXT           5245  27-04-16  19:33:13
MAN_CMD  ARC        68674  27-06-16  16:48:57
MAN_DRIV ARC       27160  18-01-15   1:34:14
      1152 FREE SECTORS
```

Please read FILES.TXT to gain information for the contents. For ARC files please use SDX own ARC command to extract the archive files.

Each program, driver or utility comes with its own documentation or a man page for the SDX online help system. When using the respective software just add the respective 'doc', 'txt' or 'man' file to your online help device and/or directory.

### 9.1 Pre-Configured Batch Files

To get a glimpse of the powerful possibilities of SDX when using batch files please try the examples in the directory BAT.

**BCREATE.BAT** - How to use counted loops.

**CARCOPY.BAT** - How to make massive batch-copy.

**UNPACK.BAT** - Unpack all ARC files found in the current directory.

**X.BAT** - Unpack the specified ARC file into a subdirectory.

**MAKE.ARC** - Facilitate assembling with MAC/65.

This SDX 4.4x batch file is supposed to act like a very simple front end for compiling programs on A8. To install MAKE.BAT just copy it to your project main directory and edit the first few lines of it accordingly. Type **-MAKE HELP** for a simple help screen.

## 9.2 Boot Loader

To be found in the directory BOOTLD.

### 9.2.1 SDLOAD - A loader for games and demos.

---

#### Purpose

Binary file loader for SDFS media.

#### Syntax

None

#### Type

External - on SDXTK disk.

#### Availability

As of SDX 4.43.

#### Remarks

Powerful successor of legacy SD boot menus like LOGOMENU or XDIR. Copy the file SD6502.SYS to a SDFS medium and utilize the BOOT command to make it bootable. Next copy the desired files to this medium and boot to the menu.

Operation: <ESC> - reload the directory from the default drive  
<A-O> - change drives  
<TAB> or <SPACE> - go to next page  
<ARROW KEYS> - move cursor  
<RETURN> - execute file (or enter dir)

Features:

- displays 66 filenames per page
- the number of directory entries that can be displayed is limited only by the amount of available memory
- the cursor position is saved when you enter a directory, so that when you come back up, the cursor is positioned over the directory name
- once SDLOAD is loaded, you can change drive number
- the loading routine is 2-3 times faster than in XDIR

**Notes:** To avoid raising MEMLO to an unacceptable value for most game files, the extended RAM is used to keep buffer. So SDLOAD requires at least 128 KiB to run. Therefore programs requiring the same extended memory area will not run.

Some newer games (Inside, Tactic) may refuse to load due to garbage left in the extended RAM. Power off for a longer while usually helps. Other games (Pole Position, Vicky) may refuse to load when the SDX cartridge is present. Remove it then.

## 9.2.2 INIDOS.SYS - restart a deactivated SDX

---

**Purpose**

Restart disabled SDX.

**Syntax**

None

**Type**

External - on SDXTK disk.

**Availability**

As of SDX 4.40 on CAR:.

As of SDX 4.43 on SDXTK.

**Remarks**

Executing the command COLD /N in the Command Processor, or causing a cold system restart while in BASIC or most application programs deactivates the SDX cartridge completely. The reactivation is usually not possible without switching the power off and back on - and this in turn, for example, invalidates ramdisks.

This problem can be solved using the INIDOS.SYS program. Copy it onto a disk, where the computer will be booted from, and then type in the command BOOT INIDOS.SYS. When SDX is about to be reactivated after COLD /N, you only need to insert this disk in the boot drive and then reboot the system (without switching the power off).

If there is no SDX available the message "SpartaDOS X not present!" will be displayed.

**Note:** The SDXTK disk is prepared to boot INIDOS.

## 9.3 Compatibility

Tools to provide backwards compatibility with legacy SD versions.

### 9.3.1 VPRINT.SYS – the respective call known from SD 3.2

---

**Purpose**

Improve backward compatibility with SD 3.2.

**Syntax**

None

**Type**

External - on SDXTK disk

**Availability**

As of SDX 4.47 on SDXTK Rev. B

**Remarks**

This program implements the VPRINT call, present in SD 3, and not implemented in SDX, where the corresponding function has been replaced by the more versatile - but not backwards compatible - PRINTF function.

## **9.4 Dynamic Link Libraries**

To be found in the directory DLL.

### **9.4.1 UNIXTIME – link library for assembler programs**

---

It is a binary library (more traditionally called an overlay) which can be used in your own assembler programs. It provides time stamp conversion functions between UNIX format (number of seconds since 1 Jan 1970) and SD format (DD:MM:YY HH:MM:SS). Time zones are not taken into account, all the stamps are interpreted as UTC.

Please read the respective text file on the SDXTK disk for details about loading, calling and functions. The S2U.COM (S2U.S MADS source) program is an example on how to use the library.

### **9.4.2 FSEL - file selector library for assembler programs**

---

This is a loadable library which can be used in your assembler programs. It provides programs with an interactive file selector functionality. EXAMPLE.S is a MADS source file which contains an example on how to invoke and use the FSEL.DLL in own program.

The selector requires about 2900 bytes of conventional RAM (above the MEMLO pointer) to run. No additional buffers for the directories are required.

Please read the respective text file on the SDXTK disk for details about installation and the capabilities provided by the library to your own assembler programs.



## 9.5 Drivers

Please look up in the directory DRIVERS

**Note:** Most drivers are fully functional but some may still be under development. Use them at your own risk.

### 9.5.1 PCLINK.SYS - link to a PC file server

---

#### Purpose

Install PCL: device allowing the DOS to act as a client of a file server using the DOS2DOS protocol. The file server may be a PC running SIO2BSD or similar peripheral emulator, the PCLink software and DOS2DOS protocol allow to access files stored directly on the file server's (PC's) disk.

#### Syntax

```
DEVICE [d:][path]PCLINK
```

#### Type

External - on SDXTK disk.

#### Availability

As of SDX 4.43.

#### Remarks

The file server must be up and there must exist a physical connection to it at the time the PCLINK.SYS is loaded. Currently the only existing compatible file server is SIO2BSD (an APE-like peripheral emulator running on BSD and Linux via the SIO2PC cable).

The driver installs device PCL: which works quite like CAR: except it is writable and recognizes subdirectories. Files on PCL: (hosted by the file server) may be then freely managed using ordinary DOS commands such as DIR, COPY, REN, DEL, MKDIR, RMDIR and so on. The device is also registered in CIO - as DPCL: and accessible from (e. g.) Atari BASIC as such.

There may be as much as 15 PCL: units, depending on how many are hosted by the file server. They're numbered the same way as disks, i. e. PCLA:-PCLO:

The driver by default uses the internal SIO drivers and UltraSpeed fast I/O protocol (which means that the file server may also be attached to the PBI bus, if any time anyone builds such a device).

The driver works best with SDX 4.43 and later.

See the batch file BACKUP and the corresponding DOC-file for creating backups via PCLink.

## 9.5.2 Screen Drivers

To be found in the directory SCRDRV.

### 9.5.2.1 S\_VBXE.SYS - use the Video board XE

---

#### Purpose

Graphics driver for the Video Board XE (VBXE).

#### Syntax

```
DEVICE S_VBXE [f,b] [/M]X
```

#### Type

External - on SDXTK disk.

#### Availability

As of SDX 4.43.

#### Remarks

The driver installs itself into the 'S:' driver of the A8 OS. Therefore all functions of the underlying system driver work as usual, but new display modes are available. Play with them easily using BASIC, or any other programming language. Alternatively, the driver can be loaded directly from the command line; it occupies only about 512 bytes of A8 main memory..

**Notes:** VBXE core FX 1.21 or later is required. See SDXLD.TXT file for details.

If the 'S2:' functionality is not required, use RC\_VBXE.SYS and save some main memory.

To access the VBXE BIOS from e. g. BASIC please refer to XIO.TXT in the SVBXE archive file for detailed information.

#### Parameters and switches

'f' and 'b' are numeric values for foreground and background color, respectively. Values given should be compatible with ones suitable for GTIA color registers; defaults are \$CA and \$94, i. e. 202 and 148.

The '/M' switch will cause the program to convert the default, 256-color Atari palette into a 16-shade gray scale palette. It is useful if a monochrome monitor is attached through the R-out and CSYNC (a 1-1.2 kΩ resistor is required to connect these two signals).

**Note:** DO NOT use this switch when running a color monitor with all the RGB signals connected.

The /X switch enables the VBXE XCOLOR mode for the GTIA display: a 256 color palette in all old modes, and in hires the pixel color is independent from the background color.

See the respective man page to read about the all new features, modes and how easy it is to use the VBXE from BASIC and Turbo-BASIC XL.

Unpack EXAMPLES.ARC to find out more about VBXE under Turbo-BASIC XL.

### 9.5.2.2 RC\_VBXE.SYS - VBXE driver using less memory

---

This is a smaller version of S\_VBXE, which does not provide the "S2:" functionality, just low level \_RAWCON interface only. It allows to save about 100 bytes of main RAM.

### 9.5.2.3 VBXE.SYS – minimalistic driver for VBXE

---

#### **Purpose**

A basic driver for the VBXE video board. It performs detection of the board and defines a symbol VBXEBASE pointing to the board's hardware registers.

#### **Syntax**

DEVICE VBXE [addr]

#### **Type**

External - on SDXTK disk.

#### **Availability**

As of SpartaDOS X 4.47.

#### **Remarks**

The address of the VBXE hardware registers may be set arbitrarily by passing it as a parameter (the "addr" one above). If no address is given, the program runs a detection routine. Currently it is only done for addresses \$D640 and \$D740. Apart from the installed symbol, the driver does not occupy memory.

### 9.5.2.4 RC\_GR8.SYS - 80-column basis

---

#### **Purpose**

Graphics 8 raw console driver to emulate a 80 column screen.

#### **Syntax**

DEVICE RC\_GR8.SYS

#### **Type**

External - on SDXTK disk.

#### **Availability**

As of SDX 4.43.

#### **Remarks**

This screen driver emulates some functions of an 80-column 'raw console' API. You should try to load the RC\_GR8.SYS, when there is no VBXE available, and while trying to load a program a message appears that alerts you about missing '\_RAWCON'.

The driver has enough functions to carry Sparta Commander, Trub Terminal for Indus CP/M and the 80-column console driver (CON.SYS).

**Note:** The driver requires the new memory management available as of SDX 4.47.

### 9.5.2.5 CON.SYS - 80-column console

---

**Purpose**

The CON.SYS file contains 80-column console, or in other words an 80-column "E:" driver. You have to load it on top of an S\_\*.SYS or RC\_\*.SYS driver to get a complete console.

**Syntax**

```
DEVICE [d:][path]CON [lmargin [krpdel [keyrep]]] [/E]
```

**Type**

External - on SDXTK disk.

**Availability**

As of SDX 4.43.

**Remarks**

The first three parameters are as follows:

- lmargin - left margin value, 0-5
- krpdel - key to repeat delay value (OS variable \$02D9)
- keyrep - key repeat delay value (OS variable \$02DA)

The defaults are inherited from the OS. You may pass only the first or the first two parameters if you like, defaults will be assumed for the rest.

To enable the 80-column console automatically when using a VBXE, pass /E switch as the last parameter.

When the screen is 80-column wide, DUMP and MDUMP will now display the memory contents in rows containing information about 16 bytes each. Other programs like Eddy will also benefit from the 80 column mode.

**Note:** To use CON.SYS on top of RC\_GR8.SYS load it inactive, then enable it via AUTOEXEC.BAT. The GR.8 display destroys the SDX startup routines in memory.

### 9.5.3 Drivers for Turbo Boards

To be found in the directory TURBOBRD.

#### 9.5.3.1 Rapidus Turbo Board MMU driver

---

##### **Purpose**

Memory management and interrupt service for the Rapidus Turbo Board.

##### **Syntax**

DEVICE RAPIDUS.SYS

##### **Type**

External - on SDXTK disk.

##### **Availability**

As of SDX 4.47

##### **Remarks**

**Memory management** - The Rapidus accelerator board is by default configured so that the CPU, when accessing the address space area of \$8000-\$BFFF, reads and writes Atari RAM rather than the Fast RAM. This is necessary if the CPU has to see cartridges (including SDX cartridge) which use this area. This however also means that this area is slow: Atari RAM can not operate at any higher speed than 1.77 MHz.

Here comes the RAPIDUS.SYS driver. When the SDX, BASIC, or any other cartridge is activated, it switches the RAM at \$8000-\$BFFF to the slow mode, so that the cartridges are usable. But when all ROMs go from there (e. g. when a program is run using X.COM), that RAM area will get switched to fast mode, so that programs may run without unnecessary penalty.

**Interrupt service** - The 65C816 processor has an extended interrupt system. This driver, therefore, installs extended interrupt vectors under the ROM, and re-installs them whenever a program is terminated, to make sure that they have not been destroyed by e. g. Turbo BASIC XL. For now, only basic IRQ and NMI vectors are installed.

##### **Make High RAM available for file I/O**

The driver also contains a patch for the SDX kernel which allows programs to read/write the High RAM (addresses  $\geq$  \$010000) without replacing all the device drivers in the system. After installing the patch, data files may be loaded to or written from the High RAM.

See details in the respective man page in the file '65816.ARC'.

### 9.5.3.2 Rapidus Turbo Board – RAM disk driver

---

**Purpose**

Ramdisk for Upper Banked RAM on Rapidus Turbo Board.

**Syntax**

DEVICE RAPIDRD [d]

**Type**

External - on SDXTK disk.

**Availability**

As of SDX 4.47

**Remarks**

This is a version of RAMDISK.SYS that uses the extra 16 MiB RAM available on the Rapidus Turbo Board to setup a ramdisk. The other 16 MiB (aka 65C816 linear memory) remain unaffected. The size of the ramdisk is fixed at 16 MiB and may get 'd' as the desired drive number.

### 9.5.3.3 65C816 RAM Speed Test

---

**Purpose**

Test relative speed of 65C816 high RAM memory access.

**Syntax**

HRAMSPD

**Type**

External - on SDXTK disk.

**Availability**

As of SDX 4.47.

**Remarks**

The program tests the relative access speed to the 65C816 high RAM. To run it, you will need a 65C816 board with high memory, an OS ROM replacement allowing to switch the CPU to native mode (Turbo-816 OS or DracOS should do), and SDX.

The program displays hex numbers in 16 columns and 16 rows. Each row, from left to right, represents 16 consecutive memory segments, 64 KiB each. After a VBL pulse each segment is read and written until another VBL pulse occurs. The number of pages, which the program managed to read/write between the pulses is then displayed in hex at the relevant position, and the program proceeds to test the next segment. So, the higher the hex number is, the faster is the memory.

If there are no more segments, the entire loop is started again: pressing any key aborts the program and quits it to DOS. Segment 0 (default 64 KiB) is never tested.

The test is not destructive, the RAM contents remains unchanged during the operation.

### 9.5.3.4 65C816 RAM Integrity Test

---

**Purpose**

Test integrity of 65C816 high RAM memory.

**Syntax**

HRAMTST [delay]

**Type**

External - on SDXTK disk.

**Availability**

As of SDX 4.47.

**Remarks**

The program tests the integrity of the 65C816 high RAM. To run it, you'll need a 65C816 board with high memory, an OS ROM replacement allowing to switch the CPU to native mode (Turbo-816 OS or DracOS should do), and SDX.

The program displays characters in 16 columns and 16 rows. Each row, from left to right, represents 16 consecutive memory segments, 64 KiB each.

The programs fills either the entire memory or individual 64 KiB segments with a random pattern. In this phase, all segments are marked with "#". After that, the program pauses for a 'delay' VBL ticks (or 192, if nothing specified), allowing dynamic RAM to fade when refreshing circuitry does not work well. Next, the memory contents is checked. When its does not match the pattern, the faulty segment is marked with "?", or with a "." otherwise.

The current pattern and the total error count is displayed at the bottom.

If there are no more segments, the entire loop is started again: pressing any key aborts the program and quits it to DOS. Segment 0 (default 64 KiB) is never tested.

The test is destructive, the RAM contents gets destroyed during the operation, so it is best to do a cold restart of the system afterwards.

### 9.5.3.5 65C816 Interrupt Driver

---

**Purpose****Syntax**

DEVICE 65816

**Type**

External - on SDXTK disk.

**Availability**

As of SDX 4.47

**Remarks**

Performs the same tasks as the RAPIDUS.SYS except memory management.

### 9.5.4 Other Drivers

These are not grouped in subdirectories, but found in the directory DRIVERS.

#### 9.5.4.1 CAD.SYS - terminate any software by keypress

---

##### Purpose

Force the return to the SDX command line from any application.

##### Syntax

DEVICE CAD [\$]keycode [\$]counter switch

##### Type

External - on SDXTK disk.

##### Availability

As of SDX 4.45.

##### Remarks

"Control + Alt + Delete" - CAD does not restart your computer, but will return to SDX from almost all programs, reopening the screen, turning off the sprites and sound, setting the low letters, margins and restoring VBL interrupts and DOSINI vector to values before installing.

Programmers sometimes forget about an exit option and lock up the <RESET> key (e. g. 'Disk Communicator 3' or 'The 1<sup>st</sup> XLEnt Word Processor'). The only thing left to do then is turn power off and start again. Well, CAD is the cure for this problem. It installs at Memlo and stays there until the next cold start or power off.

The following parameters are mandatory:

- **Keycode** is the code on which CAD will force a return to system. It may be entered in decimal or hexadecimal numbers. Choose it carefully, because e. g. 12 sends back to SDX after n-times pressing <RETURN>. Avoid key strokes used in other programs. Suggestion to use:

```
<SHIFT><CTRL><INV> ($E7, 231)
<SHIFT><CTRL><RETURN> ($CC, 204)
```

- **Counter** defines how many times the 'keycode' must have been pressed to take effect.
- **Switch** controls the Caps Lock state after the return to the command line interface.

The program confirms a successful installation or reports an error otherwise.

Of course, there are some programs that do not cooperate correctly with CAD. Hit the <RESET> key then to issue a COLD start.

**Notes:** CAD.SYS is a system application for SDX only. It may also be installed from the command line. Unfortunately, it is incompatible with OSS cartridges.



### 9.5.4.2 CPMFS.SYS - read Indus CP/M disks

---

**Purpose**

Read Indus CP/M disks.

**Syntax**

DEVICE CPMFS

**Type**

External - on SDXTK disk.

**Availability**

As of SDX 4.46.

**Remarks**

A driver to read Indus CP/M disks, i. e. the CP/M 2.2 running on Indus GT (LDW 2000, CA-2001) floppy disk drives. It is extremely useful when results under that CP/M implementation are to be copied back to A8 drives.

**Notes:** The CP/M file system uses a 'block' of sectors (also known as cluster) as a single allocation unit on a disk. Such a block is minimum 1024 bytes (4 sectors in double density, 8 sectors in single density).

For best performance, it is recommended to declare at least this many buffers for SPARTA.SYS1, plus one. I. e. 9, if using single density CP/M disks. If the number of buffers is smaller, the drive will very often (after reading each cluster, actually) seek to the directory tracks, where files are mapped. Technical details at <http://trub.atari8.info/>.

### 9.5.4.3 USER - change the user on a CP/M disk

---

**Purpose**

Changes user on a CP/M disk.

**Syntax**

USER number

**Type**

External - on SDXTK disk.

**Availability**

As of SDX 4.46.

**Remarks**

Change the user number for the CP/M 2.2 file system driver, if loaded. The 'number' may range from 0 to 15. When no CP/M file system driver is loaded, the command does nothing.

**Note:** On what the 'user' in CP/M is, please consult the CP/M documentation.

### 9.5.4.4 FATxxx.SYS - read MS-DOS filesystem

---

**Purpose**

A driver to read MS-DOS formatted media.

**Syntax**

DEVICE FATxxx.SYS (see respective version filename)

**Type**

External - on SDXTK disk.

**Availability**

As of SDX 4.46.

**Remarks**

A MS-DOS FAT filesystem read-only driver to access MS-DOS-formatted media in a limited way, provided that the I/O device is able to read disks with 512-byte sectors.

It is recommended to format floppies to 1024 bytes (2 sectors) per cluster. For hard disks the recommendation is to keep clusters at 4096 bytes (8 sectors), but in fact up to 64KiB per cluster should be possible.

There are three versions of the driver in the archive:

- FAT12A.SYS - This one only handles FAT12 media (floppies) and does not support access to subdirectories. The advantage is relatively small memory footprint. It is also the only version of the driver which does not require extended RAM.
- FAT12B.SYS - FAT12 only with full access to subdirectories. This requires one bank of ext RAM to be free at the time when the driver is loaded.
- FATFS.SYS - This one handles both FAT12 and FAT16, with subdirectories. The ext RAM is required as above. The maximum disk size is 2 GiB. SDX 4.47 or later is required.

Port B and Axlon type memory extensions are supported.

**Notes:** Reading MS-DOS (or Atari ST) formatted floppy disks may be slow because of an improper interleave for A8 and bit inversion. See the file FAT.TXT in the respective archive on SDXTK disk for more details.

These drivers enable to read the FAT partition created using the APT tools written by FJC.

### 9.5.4.5 CHKFAT – verify FAT compability

---

**Purpose**

Verify if the specified FAT disk is compatible with the FATFS.SYS driver.

**Syntax**

CHKFAT [d:]

**Type**

External - on SDXTK disk.

**Availability**

As of SDX 4.48.

**Remarks**

It sometimes happens that a disk formatted for FAT16 on a larger system is not recognized by the FATFS driver. The CHKFAT tool comes handy then and allows to avoid the trouble of examining the FAT disk boot sector values to find out, what is wrong with the disk.

CHKFAT checks the same values as does the FATFS driver. If the former displays no errors for a particular disk, the disk in question should also get accepted by the latter.

Moreover, the CHKFAT program is a bit more picky than the driver, and besides the actual problems it also checks for potential ones. So, if the driver has no problem with logging a disk, but CHKFAT finds a problem, it is strongly advisable to adjust the parameters in question and build the file system again.

The program and what is displayed should be self-explaining. Pay particular attention to the "Sector addr." parameter. If CHKFAT complains about this one, the FAT file system is too large for the used storage hardware and the whole thing is unlikely to work properly. It is recommended to use FAT disks not larger than 32 MB.

**Notes:** The cluster count will be off by 2 if compared to what 'CHKDSK /X' displays for the same disk. The real number of clusters is the one returned by CHKFAT. CHKDSK uses the internal information fetched from the FATFS driver, and the driver currently keeps the cluster number increased by 2 for convenience.

Executing CHKFAT on a non-FAT disk will return meaningless results.

### 9.5.4.6 PBI.SYS

---

The same driver as on the CAR: device. See chapter 8 for details.

### 9.5.4.7 KTRACE – trace SDX kernel calls

---

#### Purpose

Trace SDX kernel calls.

#### Syntax

DEVICE KTRACE.SYS [banks]  
 KT [command] [parameters]

#### Type

External - on SDXTK.

#### Availability

As of SDX 4.48.

#### Remarks

The program records SDX kernel calls and stores information about them in extended memory. Parameters for the calls are recorded as well, so KTRACE will provide fairly detailed information on what a particular program does when calling the system,.

The KTRACE program consists of two parts:

- the resident driver KTRACE.SYS
- the controlling utility KT.COM

KT.COM will not work, if the driver KTRACE.SYS has not been previously loaded.

KTRACE.SYS accepts just one parameter: the number of RAM extension banks to be reserved for the recording buffer. If omitted, KTRACE.SYS will allocate all banks currently available. The same happens, when the passed number is bigger than the number of available banks. During setup the number of banks actually reserved will be acknowledged on screen.

The KT.COM utility allows to control the KTRACE.SYS driver. When no argument is passed, the program will list available commands and terminate. The commands:

**INIT** this will reset the KTRACE.SYS driver to its initial state. All information held in the recording buffer will be deleted.

**DEV** with this you can limit the list of kernel devices for which the calls are being recorded. By default calls for all the devices are taken into account, i. e. DSK:, CLK:, CAR:, CON:, PRN:, NUL: and whatever is additionally installed. This however usually produces a large output filled with information which is most of the time useless. So, if you want to somewhat reduce the amount of information to be recorded, you can limit the number of the devices. For example, if you want to record calls only for the disks and the printer, do this:

```
KT DEV DSK: PRN:
```

**ON** start the recording. The program will automatically turn this to OFF when the recording buffer gets full.

- OFF stop the recording
- STAT display current status. The ON/OFF status will be displayed, a list of devices picked to be traced, and storage information, i. e. how big is the buffer and in what percentage it is full.
- DUMP will display the records in human-readable form. The output of this command looks better on an 80-column display. KT DUMP implies KT OFF.
- SAVE save the buffer to a file (for later inspection, for example).
- LOAD load the buffer from a file. The file must be generated by the KT SAVE command and must fit in the extended memory.

Use SAVE [d:][path]fname[.ext] to save and the same parameters for LOAD.

**Notes:** Additionally after each command the program outputs the number of calls having been currently recorded.

By default KTRACE records only calls directed at DSK. See DEV to add other devices.

#### **9.5.4.8 RAMD816L.SYS - SDX ramdisk for 65C816 systems**

---

##### **Purpose**

A RAM disk driver for 65C816 systems.

##### **Syntax**

DEVICE RAMD816L [drive][,nseg]

##### **Type**

External - on SDXTK disk.

##### **Availability**

As of SDX 4.43.

##### **Remarks**

This is a version of RAMDISK.SYS that allocates 65C816 linear memory (or Fast RAM) addresses past the first 64 KiB for a ramdisk.

If no drive number and/or number of 64 KiB segments are specified, SDX will apply the default settings.

In presence of the 65C816 XL OS the OS' memory handler is used to allocate the memory block required for the ramdisk. The amount of free Fast RAM decreases accordingly. The allocated block is marked 'resident' and 'RESET proof', so that it is not freed except by a cold restart. The cold restart, however, does not invalidate Fast RAM, so that loading RAMD816L.SYS (with the same size given as an argument) should restore the ramdisk contents. RAMD816L does not disable IRQs during data transfer.

The program may be used on Warp4, HyperSpeed XL/XE, F7 and XL7 accelerator boards. It also of course requires SDX, preferably 4.46 or later revision.

## 9.6 Disk Tools

Find these versatile tools in the directory DSKTOOLS. Especially when using DD512 formatted partitions these are a must.

### 9.6.1 CleanUp X - The SDX Disk Fixer

Bad errors on your hard drive? Fortunately, there is 'CleanUp X' from the SDXTK. It provides file system repairs in a professional manor.

```

A:\SDX_TK\CLX A: A\CLX A:
CleanUp X v.1.9b, (c) 2013 DLT
Sector count: 65535
Bytes/sector: 512
Analyzing boot sector...
Mapping A:>
A:>SDX_TK>RWTEST>
* Status: $88, Map: $1C05 (7173),
  Len: 0, Name: >RWTEST TST<
  The file was opened for write and
  never closed, the data is invalid.
  Delete (Yes/No/Quit)?■

```

Fig. 48: CLX 1.9b Checking DD 512 Medium

CleanUp X is a SDX-only program and requires a minimum of 31 KiB free main memory above memlo to run. The program's main purpose is to rebuild the bitmap, to reconstruct the vital part of the boot sector and to provide assistance to fix typical damages in the directories and data area. It even can repair a directory, where multiple files have the same name, but this feature requires a swap file to be setup for this purpose.

Optionally, a bitmap report showing the differences between the "old" (i. e. disk-resident) and the newly built bitmap is available. It has the form of a table being compatible with the legacy ICD CleanUp. All this enables you to inspect the relevant locations with the SDX disk editor Eddy.

SDX versions older than 4.40 require the installation of the symbol U\_GETKEY.SYS.

For detailed instructions please see the documentation on the SDXTK.

### 9.6.2 Eddy - advanced sector editor

'Eddy' was written from scratch to match the needs of a sector editor for SDX V. 4.4x. It enables the user to edit devices, disks and files and works with all media densities from SD to DD512.

Eddy will also be functional in the same manor with the Graphics 8 raw console driver RC\_GR8.SYS emulating a 80 column screen. Read the information files on the SDXTK disk for more information on how to set it up properly.

See details about the program's operation in the documentation file on the SDXTK.

```

menu Options
C: SDH443RC | 64512 (2000FC00) sectors | 2 helps
000:0003 0030 E007 4C80 3024 0000 FC77 D620
010:0400 9F02 7F00 5344 5834 3433 5243 0100
020:2100 017E 0001 674E 4022 0000 0000 0000
030:0000 0000 4572 726F 7230 204E 6F20 444F
040:539E AD28 306D 0003 AD29 303D 0B03 0900
050:022F 20F1 30AD 002F 8D28 30AD 012F 8D29
060:30A0 0484 91A4 91CC 1F30 F0D6 B900 2F8D
070:0003 B901 2F8D 0B03 C8C8 8491 606C E202
080:-----
090:-----
0A0:-----
0B0:-----
0C0:-----
0D0:-----
0E0:-----
0F0:-----

Sector: 0001 (1)      Size: 128 (0/0)      Status: 1
Format: SpartaDOS 2  Type: B00T          Allocated: Yes
UTOC: 0004 (4)      Byte: 0000 (0)      Bit: 6

```

Fig. 49: Edit Screen Eddy 2.01

'Eddy' is intended to be used with SDX in the first place, but should work with other DOS too. See the TXT-file on the SDXTK disk for details when not using SDX.

**Note:** For convenience put DEVICE COMEXE<sup>92</sup> in your CONFIG.SYS when using Eddy.

### 9.7 TSR Programs for SDX 4.2x

TSRs are some help to enable SDX versions before 4.40 to access features known from later SDX versions. They are found in the directory SDX42

ERR\_GMSG.ARC - enables detailed descriptions for all the error codes.

U\_GETKEY.ARC - supplies the missing U\_GETKEY symbol for older SDX versions. The presence of the symbol is necessary to run such programs as e. g. Cleanup X.

XCOMLI.ARC - supplies the missing XCOMLI symbol for older SDX versions. The presence of the symbol is necessary to run some programs written for later SDX versions.

JFSYMBOL.ARC - supplies the missing JFSYMBOL jump for older SDX versions. JFSYMBOL (at \$7EB) allows to get an address of a SDX symbol from a non-SDX binary program.

<sup>92</sup> Chapter 8 - Application Drivers.

## 9.8 Shells

Look-up the directory SHELLS.

MYDUP.ARC - MyDOS-alike shell.

SC100.ARC - Sparta Commander, a Norton-alike shell for SDX.

### 9.8.1 MyDUP

This is a MyDOS-like shell for SDX (V. 4.42 or newer one is required). It mimics the behavior of the MyDOS menu. There are of course slight differences due to different nature of both DOS types, but if you're familiar with the MyDOS, you should quickly find them out.

Since MyDUP uses environment space to keep some of its settings, it is recommended to install the ENV.SYS<sup>93</sup> extension to prevent filling the environment up completely.

### 9.8.2 Sparta Commander

This is V. 1.00 of the Sparta Commander. It requires SDX 4.47 or newer and a 80-column text display.

```

SpartaDOS X 4.48 | Wed 16-Mar-16 | 17:45:44
>:~*,*
<DIR> 9-03-16 21:43
SPARTA  DOS ... <DIR> 9-03-16 21:44
AUTOEXEC BAT ... 54 9-03-16 21:47
COM     SYS ... 3024 29-07-15 14:37
RC_GR8  SYS ... 3247 16-07-13 19:30
CHANGES TXT ... 10441 28-02-16 11:30
INSTALL TXT ... 3144 19-08-15 19:30
SC      COM ... 1210 21-08-15 17:30
SC      IMI ... 2539 25-02-16 13:08
SCBATHAN OVL ... 946 15-05-15 22:33
SCRAIN  OVL ... 12790 28-02-16 11:51
137 KB Free
163
help Ctrl+Alt Copy Del Edit Trash c:mod n:dir Log Move Quit Rename Tag View EXEC
  
```

Fig. 50: Sparta Commander 1.00

Since Sparta Commander uses environment space to keep some settings, it is beneficial to install the ENV.SYS extension. It is advisable to use COMEXE.SYS<sup>94</sup> as well.

Though it is recommended to have a VBXE (with FX core 1.21 or later) installed for speed reasons, Sparta Commander will also be functional in the same manor with the Graphics 8 row console driver RC\_GR8.SYS emulating a 80 column screen. Read the information files on the SDXTK disk for more information on how to set it up properly.

<sup>93</sup> Chapter 8 - Other Drivers.

<sup>94</sup> Chapter 8 - Application Drivers.



## 9.9 Utilities

All to be found in the directory UTILS covering a big variety of tasks.

### 9.9.1 APETIME - get time & date from APE

---

#### Purpose

To get the current date and time from APE (or a compatible peripheral emulator program) and store it into SDX date and time registers.

#### Syntax

DEVICE [d:][path]APETIME [/Q] or APETIME [/Q]

#### Type

External - on SDXTK disk.

#### Related

DATE, TIME

#### Remarks

APETIME.SYS may be executed at startup by adding it to your CONFIG.SYS or at any time by typing its full filename at the DOS prompt. For the latter case it may be useful to put APETIME.COM somewhere within your \$PATH\$, or make it RAM-resident with LOAD APETIME.COM.

Upon successful retrieval of the date and time value, a message is printed displaying the current time. Add the /Q switch to the call to suppress the message.

When the retrieval fails, the program will print an error message.

In conjunction with JIFFY.SYS<sup>95</sup> this provides a substitute for a hardware clock, when you have none.

**Note:** Using a PC connection with APE or AspeQt 0.7.x will download time and date from it and set the respective Atari clock.

AspeQt versions 0.8.x and higher do no longer support this feature. Please use ASPECL.COM instead provided with the AspeQt download package.

RespeQt R3 supports both APETIME and ASPECL.

---

95 Chapter 8 - JIFFY.SYS Driver.

## 9.9.2 CDD - change drive and directory

---

### Purpose

Change the current working drive and directory to the specified ones.

### Syntax

CDD [d:][path]

### Type

External - on SDXTK disk.

### Related

CD, PWD, PHD, PLD

### Availability

As of SDX 4.47.

### Remarks

The command works similarly to the CHDIR command, except that it also changes the current drive to the specified one.

## 9.9.3 CRC32 - compute a cyclic redundancy check

---

### Purpose

Compute a 32-bit cyclic redundancy check (CRC-32) for each given file.

### Syntax

CRC32 [/H] [d:][path]filename.ext [...]

### Type

External - on SDXTK disk.

### Availability

As of SDX 4.45.

### Remarks

The program writes to the standard output three space-separated fields for each input file. These fields are:

- the checksum CRC-32,
- the total number of bytes in the file,
- and the file name.

The CRC-32 value is typically used to ensure that files transferred by unreliable means have not been corrupted.

The results produced are compatible with the "cksum -o 3" command in Unix. When the /H switch is given, the CRC-32 checksum is printed as a hexadecimal number, or a decimal otherwise.

### 9.9.4 DD - derive data

---

**Purpose**

Copy a number of bytes from stdin to stdout.

**Syntax**

DD [skip [count]]

**Type**

External - on SDXTK disk.

**Availability**

As of SDX 4.43.

**Remarks**

Utility that allows copying parts of files. The command skips a number of bytes and copies a number of bytes from stdin to stdout. It can be useful to:

1. Create files of defined size and filled with zeros. For example:

```
DD 0 256 <<NUL2: >>ZEROS.DTA
```

This will create a file named ZEROS.DTA, being 256 bytes in length and filled with bytes of zero value.

2. Create files of defined size and filled with random bytes. For example:

```
DD 0 128 <<NUL3: >>RANDOM.DTA
```

This will create a file named RANDOM.DTA, being 128 bytes in length and filled with random values (check it with DUMP command).

3. Copy a number of bytes from one file to another. For example:

```
DD 1000 2000 <<SRC.DTA >>DEST.DTA
```

This will copy 2000 bytes from SRC.DTA to DEST.DTA starting from the 1000<sup>th</sup> byte.

4. Copy data from a file skipping a number of its leading bytes. For example:

```
DD 1000 <<SRC.DTA >>DEST.DTA
```

This will copy data from SRC.DTA to DEST.DTA skipping first 1000 bytes.

5. Extract a part of a file. For example:

```
DD 6 40 <<JSETNEO.NEO
```

This will extract and display the name of a NeoTracker module.

It is possible to use DD in a pipeline, like this:

## TYPE JSETNE0.NEO | DD 6 40

This extracts and displays the name of a NeoTracker module using a different method.

The usage information is displayed, when DD is executed without parameters.

### 9.9.5 DELDUP - delete duplicates

---

#### **Purpose**

Compare given directories and delete, or warn about, duplicated files.

#### **Syntax**

```
DELDUP [d:]path1> [d:]path2> [/RV]
```

#### **Type**

External - on SDXTK disk.

#### **Remarks**

The program scans the given directories and, if it finds out, that both contain copies of the same file, it displays the full pathname of the one situated in the first directory, having it preceded with "\*\*\* Dup: ".

If /V (verbose) is given, the program first displays full pathnames of both files in question. The "\*\*\* Dup: " comment is next displayed when it has been determined that both files are exact copies of each other.

Finally, if /R (remove) is given, the program will delete the file found in "path1", and tell you about it preceding its pathname with "\*\*\* Del: ".

The program is useful, when you have some 1000-odd games/demos/whatever on your hard disk, and have done a massive copy of another game/demo/etc. collection from another hard drive belonging to a fellow Atarian.

Say, your collection is in GAMES> and the newly copied stuff is in FELLOWS> - now when you do this:

```
DELDUP FELLOWS> GAMES> /R
```

the program will scan the FELLOWS directory and delete everything you already have in GAMES.

#### **WARNINGS:**

- The program does NOT scan directories recursively.
- The program skips directories, hidden files and protected files - so, before the scan, make sure all files in both directories are unhidden and unprotected.
- The program considers two files different even if they differ only by one byte.

### 9.9.6 DPOKE - double poke

---

**Purpose**

Puts a two-byte value (a word) into the given address.

**Syntax**

DPOKE [[ $\$$ ]index:][ $\$$ ]location [ $\$$ ]value[,1]

**Type**

External - on SDXTK disk.

**Related**

POKE, PEEK

**Availability**

As of SDX 4.40 on CAR: device.

As of SDX 4.45 on SDXTK.

**Remarks**

DPOKE allows to change memory locations from the command processor, which can be useful in batch files and other applications. It is very easy to crash the system with this one if you do not understand what you are doing. Using DPOKE needs to preserve the common 6502 byte order (low/high).

DPOKE also accepts memory index like the internal commands PEEK and POKE<sup>96</sup> and can even poke a single byte (lower) if a value of 1 follows the value.

### 9.9.7 DU – disk used

---

**Purpose**

Calculate the size of the specified directory.

**Syntax**

DU [/Q[QQ] [+|-][AHPS] [d:path]

**Type**

External - on SDXTK disk.

**Related**

DF

**Availability**

As of SDX 4.48 .

**Remarks**

The command scans the specified directory recursively, then outputs information about the total size of all the files and subdirectories it had encountered.

---

<sup>96</sup> Chapter 4 – Commands PEEK and POKE.

While scanning, the program displays the directory entries being processed. When the switch /Q is specified, file entries are not displayed (but they are still counted). Specifying /QQ suppresses all the display output except the final summary.

## 9.9.8 FA\_REG - Fast Assembler - registered version

---

FA is a SDX-only assembler, which has the ability to create binary sets for SDX. Start it from the command line and specify the names of the source and destination files as parameters.

## 9.9.9 FSTRUCT - display file structure

---

### Purpose

Analyze a binary file.

### Syntax

FSTRUCT [+A|H|P|S] [-A|H|P|S] [d:][path]filename.ext

### Type

External - on SDXTK disk.

### Availability

As of SDX 4.40 on CAR: device.

As of SDX 4.45 on SDXTK.

### Remarks

The command displays the information about the structure of the specified binary file (type, load address, length of the segments etc.).

## 9.9.10 HDSC - hard disk sector copier

---

### Purpose

Sector copier for large partitions.

### Syntax

hdsc [options] ds: dd:

### Type

External - on SDXTK disk.

### Availability

As of SDX 4.43 on SDXTK.

### Remarks

This is a very simple sector-copy program for SD only. All data transfers are done via the OS. Therefore partition backups by mirroring the entire contents to any legal storage device are possible.

The program should accept all source/destination combinations which match the following rules:

- Sector size is the same on both disks.
- The source disk is not bigger than destination.
- Both disks recognize PERCOM configuration commands.
- Source and destination disks are not the same drive.

Some of these limitations can be overridden with command line options.

'ds:' and 'dd:' are SD source and destination drive specifications, respectively.

Options: -C=n process 'n' sectors  
 -B batch mode (no questions)  
 -I ignore destination size  
 -L do not use SpartaDOS SIO (get BPS from source disk)  
 -2 force 256 BPS on source disk  
 -S switch off display  
 -V verify instead of copying

Except for -C, all options can be put together, e. g. -BLS is perfectly valid.

**Notes:** To use the program you need SD, because it has a command line interface.

For convenience put DEVICE COMEXE<sup>97</sup> in your CONFIG.SYS when using HDSC.

### 9.9.11 HEXDEC - hex-dec converter

---

#### **Purpose**

Convert values to hex and dec on the command line.

#### **Syntax**

HEXDEC [number]

#### **Type**

External - on SDXTK disk.

#### **Availability**

As of SDX 4.47.

#### **Remarks**

The program writes to the standard output a hexadecimal and a decimal equivalent of the given value. A hex number given must be preceded with a "\$" sign. The conversion range is 0-4294967295 (\$0-\$FFFFFFF).

### 9.9.12 INVERSE - invert input

---

**Purpose**

An example program to demonstrate pseudo-pipelines.

**Remarks**

The program receives input from a pseudo-pipeline, inverts it, and sends out to another pseudo-pipeline. To see the effect, try this:

```
ECHO The quick brown fox | INVERSE | MORE
```

### 9.9.13 LS - list files

---

**Purpose**

List the files present in the specified directory.

**Syntax**

```
LS [/C] [d:][path>][fname.ext]
```

**Type**

External - on SDXTK disk.

**Related**

DIR

**Remarks**

The program lists the files existing in the specified directory. When no directory is specified, the current one is listed.

The file (and path) names are by default listed in a SD-compatible form, i. e. with drive letters, or without them, if current drive is referenced.

Adding /C switch produces a list of AtariDOS-compatible pathnames, i. e. prefixed with "D:" or "Dn:" as necessary.

The purpose of the program is produce a list of files for another program, which can accept such a list as a parameter. For example, the D2D player, which can playback WAV samples from hard disk, is able to accept a file that contains a list of files to playback. Therefore it may also be fed from LS.COM via pseudo-pipeline, this way:

```
ls /c | d2d -p -
```



### 9.9.14 MACH - machine details

---

**Purpose**

Display hardware information which is relevant to SDX.

**Syntax**

MACH

**Type**

External - on SDXTK disk.

**Availability**

As of SDX 4.47.

**Remarks**

The program outputs the following information:

- Hardware type: 400/800 or XL/XE
- CPU type: 6502, 65C02 or 65C816
- Address space: 64 KiB for 6502/65C02 or 16 MiB for 65C816
- Clock speed in percents - standard is 100%
- Available memory: sizes of conventional, extended and high RAM.

That last type of RAM is available on computers equipped with 65C816. Additionally the first available address of the high RAM is displayed.

Extended RAM information will also include its type (PORTB or Axlon).

The program does not do any testing, it fully relies on the information SDX collects and holds internally for own purposes.

### 9.9.15 MD5 – hash tool

---

**Purpose**

Compute a 128-bit MD5 hash for each given file, or verify if the given hashes match the given files.

**Syntax**

MD5 [/H] [+|-][A|H|P] [d:][path]filename.ext [...]

MD5 /C [+|-][A|H|P] [d:][path]hashfile.md5

**Type**

External - on SDXTK disk

**Availability**

As of SDX 4.48

**Remarks**

The program writes to the standard output the MD5 hash value for each given file. The hash is written out in hex. It is preceded by an equal sign, the name of the file included in round brackets, and the text "MD5". For example:

```
MD5 (FOO.BAR) = 16ACE8916465CCDF1DC1F97BD5F342F4
```

The line is terminated by an EOL character (ASCII 155 or \$9B).

You can redirect this output to a file by standard means.

Alternatively, if the /C switch is given, the program expects a single specification of a file containing hashes. Inside, the hashes should be stored in the consecutive lines, in the format as above, or in the format produced by the md5sum tool for Linux.

The lines may be separated with an EOL character (ASCII 155 or \$9B - Atari format), an LF character (\$0A - Unix format) or a CR/LF sequence (13/10 or \$0D/\$0A - DOS format). The program will try to open files named there, calculate the MD5 hash value for each one, and match it with the hashes stored in the file. It then outputs information, if the hashes do match or they do not.

The /H option switches off the display during execution, to speedup the calculations.

You can expect that a 16 KiB file will be hashed in about 7 seconds (or about 5, if the /H option was given). See also the compile for the 65C816 processor MD5\_816.COM on SDXTK.

### 9.9.16 MEMINFO – memory information

---

#### **Purpose**

Displays technical information about the extended (banked) RAM.

#### **Syntax**

MEMINFO

#### **Type**

External - on SDXTK disk

#### **Related**

MEM

#### **Availability**

As of SpartaDOS X 4.48

#### **Remarks**

The command displays internal information about the extended RAM: which type of extension is in use (Axlon or PORTB), what is the total bank count and the free bank count etc.

The most interesting information may be the list of banks currently occupied and free. All banks are listed, and both the memory index assigned to the bank and its corresponding register value are displayed as used internally by the SDX. It may help configuring programs which allow to assign arbitrary bank numbers for their own use - now you can know which banks are already allocated by SDX and avoid a conflict.

### 9.9.17 MKATR - make an atr image

---

#### Purpose

Create a disk image file in ATR format.

#### Syntax

MKATR [d:][path]filename.ext [/DQBS] [size][,fill]

#### Type

External - on SDXTK disk.

#### Remarks

The command creates an ATR disk image file. The default geometry is SS/SD, i. e. 720 sectors, 128 bytes each.

Options:

- /D - sector size 256 bytes
- /Q - sector size 512 bytes
- /B - double density, boot sector size 256 bytes
- /S - initialize SDFS 2.1 on existing image

Parameters:

- size - total sector count (default is 720)
- fill - a value to fill empty sectors of the image (default is \$00)

#### CP/M disk images

The fill pattern parameter is useful when creating ATR images to be used with Indus CP/M. Setting the fill pattern to \$1A creates an image directly writable from the CP/M command processor. Example:

```
MKATR CPM_SD.ATR,,$1A
```

creates an SD image, and:

```
MKATR CPM_DD.ATR /D,,$1A
```

creates an DD image. Note that disks created this way are writable, but not bootable! You can only use them as disk B: or C: under Indus CP/M.

### 9.9.18 NVWGL & NVPEEK - inspect and manipulate NV-RAM

---

These programs allow to peek/poke non-volatile memory located on a real-time clock hardware. NVPEEK is an ordinary PEEK/POKE-like command, whereas NVWGL is a more interactive program which allows to see the entire clock's memory at once.

There are some requirements:

- there must be a real-time clock hardware attached to your Atari;
- the clock must be equipped with NV-RAM (some are not);
- the appropriate clock driver must be loaded.

Currently IDE+2.0 clock and ULTIMATE clock drivers allow these two programs to inspect and manipulate NV-RAM.

Remember, however, that NV-RAM memory is most of the time used and the contents may be private to the board which contains the clock (it is certainly so on IDE+2.0). Random fiddling with the values may cause undesired results; f.e. on IDE+2.0 it will most probably lead to the board's settings being lost.

### 9.9.19 PHD - push directory

---

#### **Purpose**

Stores the current directory for use by the PLD command, then changes to the specified directory.

#### **Syntax**

PHD [d:path]

#### **Type**

External - on SDXTK disk.

#### **Related**

CD, PWD, PLD

#### **Availability**

As of SDX 4.44.

#### **Remarks**

The command works similarly to the CD command, except that it stores the current path. You can then return to this directory using the PLD command.

Only one directory can be "saved" this way, using PHD twice overwrites the previously saved path.

### 9.9.20 PLD - pull directory

---

**Purpose**

Changes to the directory path stored by the PHD command.

**Syntax**

PLD

**Type**

External - on SDXTK disk.

**Related**

CD, PWD, PHD

**Availability**

As of SDX 4.44.

**Remarks**

The command restores the path stored previously by the PHD command. When no PHD command was executed previously, the PLD does nothing.

### 9.9.21 RDDUMP & RDLOAD - ramdisk save & restore

---

**Purpose**

Save and restore ramdisk contents.

**Syntax**

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

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

**Type**

External - on SDXTK disk.

**Related**

RAMDISK.SYS

**Availability**

As of SDX 4.42 on CAR: device.

As of SDX 4.45 on SDXTK disk.

**Remarks**

These two commands allow the user to save the ramdisk content into one file on a real storage device before powering off, and restore it with the next system start to use it again. This is helpful when using storage devices that are much slower than ramdisks. At system startup the dumped contents will be reloaded into memory; the ramdisk works as a kind of volatile hard drive.

Configure the system to automatically install a ramdisk during bootup. Setup directories and copy files to it as appropriate (e. g. your preferred tools & utilities). When finished dump the ramdisk to a file. See the example:

```
D1:RDDUMP 0: C:>SYSFILES>RAMDUMP.BIN
```

Use the RDLOAD command to restore the contents of the ramdisk from this file. Omit the second parameter to get it automatically from the environment variable \$RAMDISK (which is set by RAMDISK.SYS during boot time). Example:

```
D1:RDLOAD C:>SYSFILES>RAMDUMP.BIN
```

This operation requires two conditions to be met:

- The destination drive must be identified as ramdisk , and
- its capacity has to match the size of the dump file.

To minimize the time required to restore the ramdisk from a slow storage device, RDLOAD only reads sectors, which at the time of creating RDDUMP were marked as occupied. This is worthwhile when dumping a ramdisk filled to ca. 50%. To restore a nearly filled one consumes eventually more time than a simple sector copy.

To avoid restoring the ramdisk, when it was not reformatted after the last restart, edit the AUTOEXEC.BAT file and put the following lines into it:

```
IF RAMDISK
  IF NOT EXISTS $RAMDISK>*. *
    RDLOAD C:>SYSFILES>RAMDUMP.BIN
  FI
FI
```

Of course, you can specify the name of a characteristic file (or directory - in this case +S is required after EXISTS) instead of \*.\*.

## 9.9.22 RPM - check rotation speed

---

### Purpose

To check the RPM of a floppy disk drive.

### Syntax

```
RPM [d:]
```

### Type

External - on SDXTK disk.

### Availability

As of SDX 4.45 on SDXTK.

### Remarks

This command will continuously check and display the number of revolutions per minute (RPM) made by a drive until any key is pressed. This is mainly useful as a diagnostic tool to determine if a floppy disk drive is operating at the proper speed (288 RPM for most A8 drives, including the 810 and 1050, and 300 RPM for the XF551). A standard 5¼" floppy disk drive as a slave drive for a TRAK-AT will also show 300 RPM, since this is the standard speed for those drives.

This command will also return the RPM information for a hard drive.

Using RPM on PBI IDE controllers like the MSC, which are hosting CF cards, provides the relative access speed.

Using RPM on a ramdisk will simply provide the relative access speed of the ramdisk, a worthless but interesting piece of information. It will not work properly on enhanced floppy disk drives while track buffering is enabled, either.

### 9.9.23 RUN - run the code

---

#### **Purpose**

Run the code at the specified address.

#### **Syntax**

RUN symbol | [[ $\$$ ]index:][ $\$$ ]address

#### **Type**

External - on SDXTK disk.

#### **Availability**

As of SDX 4.40 on CAR: device.

As of SDX 4.45 on SDXTK disk.

#### **Remarks**

The command makes a JSR to the specified address. It is not sanity-checked before, it is assumed, that the user is invoking the command on purpose and is sure what he is up to.

As of SDX 4.48 the command has more flavours:

- 1) it supports symbols, you can give a symbol name as an argument, and, if the symbol exists, it will get translated to an address and jumped to.
- 2) it supports extended memory: you can specify the memory index of an extended memory bank you want the command to do the call.
- 3) it supports 65C816 and its 24-bit address space: you can specify a 24-bit address you want to be jumped to.

Calls made within the 6502 address space ( $\$0000$ - $\$FFFF$ ) are done using the "short" JSR instruction. It is expected, that the called routine ends with an RTS.

Calls made to addresses past the  $\$FFFF$  are done using the "far" JSR instruction (or JSL) and the CPU is in native mode. It is expected, that the called routine ends with an RTL. The interrupts are off when the underlying OS does not provide native interrupts services - otherwise they are on.

### 9.9.24 SL - symbol list manipulation

---

#### Purpose

Manipulation of a list of SDX symbols.

#### Syntax

SL [name [[ $\$$ ]address]]

#### Type

External - on SDXTK disk.

#### Availability

As of SDX 4.40 on CAR: device.

As of SDX 4.45 on SDXTK disk.

#### Remarks

'SL' invoked without any parameters will list all symbols.

Name	Address	Magic
=====	=====	=====
_RDIO	\$0FA5	\$86
I_SETTD	\$0F2D	\$85
I_GETTD	\$0F30	\$85
SIOPARM	\$692C	\$83
FILE	\$5935	\$81
...		

Typing a number of initial characters of the symbol's name will list matching symbols only. For example 'SL U\_' will show all symbols starting with U\_ like:

Name	Address	Magic
=====	=====	=====
U_LOAD	\$A269	\$00
U_UNLOAD	\$A25B	\$00
U_FAIL	\$A669	\$00
...		

Additionally, new symbols may be added to the list. E. g. a command like:

```
SL COVOX $D600
```

will add a symbol named "COVOX" with a value of \$D600. Please remember that a symbol may not carry values bigger than 65535.

**CAUTION:** This function should be used with extreme care, because adding symbols with arbitrary values may severely affect the stability of the system. This function has been implemented for testing software in development and such occasions.



### 9.9.25 STAT - status information

---

**Purpose**

Display a more detailed information on a file or directory than the DIR command does.

**Syntax**

STAT [+AHP] [d:][path]filename.ext

**Type**

External - on SDXTK disk

**Availability**

As of SDX 4.46.

**Remarks**

The program displays the following information about the given file or directory:

```
Name: [STAT .MAN]
Attr: -----
Node: $B936/47414 on DSK3:
Size: 1020 bytes (2 sectors)
Date: 17-02-16 Time: 13:34:38
```

- Name: it's file name in square brackets
- Attr: attributes, if set. The attributes are: XSAHP. The latter four ones are the standard Subdirectory, Archived, Hidden and Protected. The first one, eXtended, is reserved for future use.
- Node: a number uniquely identifying a file on the disk. What it exactly is, it depends on the file system. Usually it is the number of the first sector allocated for that file (not necessarily for its data, though).
- Size: the exact size in bytes, and an estimated size in sectors (note that not all file systems allow to calculate this value exactly).
- Date/Time: date and time of the last modification.

### 9.9.26 TAR - archiver for SD

---

#### Purpose

Create and maintain file archives.

#### Syntax

TAR command[option] tarfn [flist|fmask]

TAR command[option] [d:]device[path]tarfname.ext] [filelist|mask]

#### Type

External - on SDXTK disk.

#### Availability

As of SDX 4.43.

#### Remarks

SD TAR resembles the well-known Unix archiver TAR and enables the user to create or unpack Unix compatible archives. SD TAR is able to archive and unpack entire directory trees, but does not compress or decompress. It serves SDX as well as SD.

**Note:** Make sure that no pathname included in the tar archive exceeds 63 characters<sup>98</sup>.

Commands:        -c create archive  
                  -t list archive contents  
                  -x extract from archive

Options:           -k don't overwrite existing files  
                  -m don't restore modification time  
                  -v verbose operation

"tarfn" is TAR archive file name. "flist" is a list of directories or files to add. "fmask" is a mask for files to be listed or extracted: "\*.\*" is assumed, if no parameter is given.

#### Create an archive

```
tar -c foo c:>bar> d: e:>*.exe
tar -cv foo c:>bar> d: e:>*.exe
```

This creates a file called foo.tar in the current directory. It will contain files and directories found on paths listed.

When no file mask is specified, \*.\* is assumed. The mask is only for files, the program always archives all directories it finds, regardless of the mask. Of course, if files inside the directory don't match the mask, the directory will be archived as empty.

To archive a subdirectory c:>foo> it has to be terminated with the path separator. If just c:>foo is given, TAR will try to find a file called 'foo' and archive it.

**Note:** Directory names need to be fully written. Wildcards are not allowed here.

### To list the archive contents

```
tar -t foo *.obj
tar -tv foo *.obj
```

All filenames are listed which match the mask. Directories are only listed when they match the mask.

The `-v` option makes the program to list additional information. This will be:

- modification time in unix-like format; day, month and time are displayed for more recent files, and day, month and year are displayed for files older than 6 months (specifically; older than 15778800 seconds).
- file length for regular files or short information about entry type for all others (for a subdir it will say "<DIR>").
- file name with full path (as stored in the archive).

TAR archives usually contain more information, but under SDX all that is irrelevant and will be silently ignored.

### To extract files from archive

```
tar -x foo *.obj
tar -xv foo *.obj
```

This function extracts all files and directories with filenames matching the mask. Directories which don't match the mask will be only recreated on the disk if they contain files that do. The rest of the available options is self explaining, more or less. After typing `tar` without arguments at the DOS command prompt the description will be displayed.

### Illegal file names

On a PC it is easy to make an archive which contains file names illegal to SDX. Please see the DOC file for TAR on the SDXTK disk for comprehensive information.

**BUG:** Unpacking the same archive a second time to the same directory, without deleting the previous contents, doubles the number of resulting files.

**Note:** Please see the DOC file on the SDXTK disk for more detailed information on TAR.

### 9.9.27 TREE - display directory tree

---

**Purpose**

Displays all the directory paths found on the medium or under the specified directory, and optionally lists the files found in each directory in alphabetical order.

**Syntax**

```
TREE [d:][path] [/F]
```

**Type**

External - on SDXTK disk.

**Availability**

As of SDX 4.45.

**Remarks**

The TREE command displays all path names found on the diskette when used from the main directory. If a path is specified, then all pathnames under that directory will be displayed. When used from a subdirectory, TREE will display all path names from that directory on. If the '/F' is specified, then all filenames in each directory will be displayed in alphabetical order after the directory path they're in.

Example: TREE D1:MODEM /F

Subdirectory MODEM is displayed as the root directory and all filenames under that are displayed; then any subdirectories under MODEM are displayed along with the filenames under each of those. This continues until the last subdirectory and filenames are displayed.

### 9.9.28 TSR.SYS - enhance the number of TSR programs

---

**Purpose**

Extend the system extension mechanism also known as TSR (Terminate and Stay Resident).

**Syntax**

```
DEVICE TSR [size]
```

**Type**

External - on SDXTK disk.

**Availability**

As of SDX 4.47.

**Remarks**

Allow to register much more number of TSRs with S\_ADDIZ routine and INSTALL flag (standard SDX extension mechanism allows to register only 5 TSRs). Procedures registered by TSR.SYS are numbered from 5 (index of routine is placed in FLAG when it is initialized).

'Size' is the maximum count of TSRs possible to register (16 by default, 251 at max).

### 9.9.29 WC - words counter

---

**Purpose**

Counts the number of words, lines and characters in a text file.

**Syntax**

WC [/W]/[L]/[C] [filename]

**Type**

External - on SDXTK disk.

**Availability**

As of SDX 4.44.

**Remarks**

This command resembles a command for Unix-like operating systems. It generates the following statistics for a file: number of words, lines (specifically, the number of EOL characters) and bytes (characters). If no filename is given, the command uses standard input. Option /W prints the word count, /L prints the line count, /C prints the byte count. When no option is given, all three values are printed.

**Examples**

WC FILE.TXT - print word, line, char count and the filename

WC /W FILE.TXT - print word count only

WC - CON: is used to enter text, press <CTRL><3> to stop and get the counts

DIR | WC /L - print number of lines in a directory listing

COMP FILE1.TXT FILE2.TXT | WC /L - print the number of differences between these files

### 9.9.30 XFCONF - density selection for floppy disk drives

---

**Purpose**

Select the density in floppy disk drives.

**Syntax**

XFCONF [d:] [/12345]

**Type**

External - on SDXTK disk.

**Availability**

As of SDX 4.40.

**Remarks**

Some floppy disk drives have problems when it comes to automatic density detection. This problem particularly beats the Atari XF551 disk drive (thus the name of the command), but the program can be handy in case of any other drive too, when it happens to be somehow stuck in an improper density. Then, the XFCONF allows to force the drive manually into the proper one.

The command invoked without arguments displays menus, where you can choose drive number (1-4) and the desired density. After the new configuration was sent out to the

drive, it is checked, whether the drive has really selected it (some drives do accept and positively acknowledge densities impossible for them, for example 360 KiB for an Atari 1050 Top Drive, or 720 KiB for an Atari XF551 - the drive's controller selects something closest to the requested configuration and replies "OK" to the computer - which is surely far from being OK). The message Drive cannot do this density means, that the requested density was accepted, but the drive is in fact unable to realize it.

### **Other messages**

Drive not configurable - The drive is a stock Atari 810 or Atari 1050, no modifications. Density changes are not possible.

Drive is not a floppy disk - An attempt to change the density of a ramdisk or hard disk has been tried.

Drive rejected the density - The drive explicitly denied to select the density.

XFCNF accepts command line arguments too. Giving a drive identifier alone causes the computer to check, if the drive is a floppy drive and if it is configurable. The reply Drive is configurable confirms that. The density change can be accomplished by adding a switch followed by a digit from 1 to 5, where the digits mean densities as follows:

- /1 - single density (SS/SD, 90 KiB)
- /2 - dual density (SS/ED, 130 KiB)
- /3 - double density (SS/DD, 180 KiB)
- /4 - double sided double density, 40 tracks (DS/DD, 360 KiB)
- /5 - double sided double density, 80 tracks (DS/DD, 720 KiB)

## **9.9.31 XT - execution time**

---

### **Purpose**

Display the execution time of a given command.

### **Syntax**

XT command [parameters]

### **Type**

External - on SDXTK disk.

### **Availability**

As of SDX 4.44.

### **Remarks**

The program will count the execution time for a given command line and display the result upon command termination.

The time measurement is VBL-based, so if the command line executes an operation that involves disabling the system VBL routines, the displayed result will not be correct.

Note that in case of executing programs, the loading time also will be added. And even with that counted in, the result will still be slightly (about 0.5-0.7 second) off due to

command line execution overhead. Try a command that "does nothing" (such as XT CLR) to see what the overhead is.

The XT command requires SpartaDOS 4.44 or a later revision.

### **9.9.32 XVER - SDX build version**

---

#### **Purpose**

Display the current version number, date of the cartridge and build type.

#### **Syntax**

XVER

#### **Type**

External - on SDXTK disk.

#### **Remarks**

This command will show the version number, revision date, and cartridge type.

### **9.10 Man Pages**

As excerpts from this guide man pages for SDX own online help system.

MAN\_CMD.ARC - A collection of manual pages for SDX commands.

MAN\_DRIV.ARC - A collection of manual pages for SDX drivers.





## 10 SpartaDOS X Add-Ons

Download these specials from the project page to enhance the SDX environment.

### 10.1 By DLT

#### 10.1.1 SDX Imager

Current version 3.0.6 - a Windows utility for advanced users to customize CAR: contents; e. g. add/remove user files, modify default CONFIG.SYS etc.

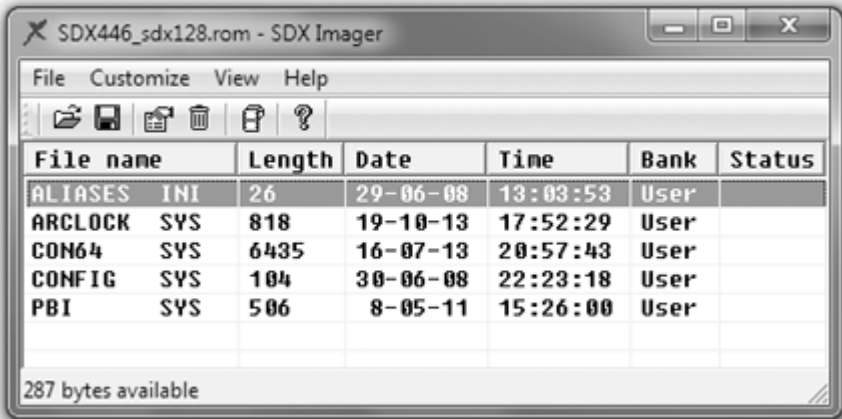


Fig. 51: SDX Imager - User Bank Content

#### 10.1.2 SDX2Atrax converter

Windows command-line utility to convert between intSDX128 and Atrax images.

#### 10.1.3 MUXTIME for SDX

A tool to retrieve time and date settings from the Multiplexer! master machine.

#### 10.1.4 Emu-pack

Current version 0.2 - access emulator's "H:" device from COMMAND.COM. Allows to check the directory and to copy files from/to the "H:" device.

#### 10.1.5 S2I - a SIO2IDE-specific tool

Enables the user to address a connected SIO2IDE device with firmware 3.x and 4.x.

#### 10.1.6 S2S - a SIO2SD-specific tool

Enables the user to address a connected SIO2SD device.

#### 10.1.7 MNT - a KMK/JZ IDE 1.0 / IDEa specific tool

Enables the user to address a connected KMK/JZ IDE or IDEa device.

**Note:** Do not use with IDE Plus 2.0!

## 10.2 Contributed by Bober

### 10.2.1 Aprplr - APR animation player

Experimental animation player for SDX using the APR format. It uses any GTIA mode with resolution at 80 x 48 and a differential compression, horizontally or vertically, depending which is more suitable.

### 10.2.2 Core Wars - SDX Core Wars package

Enjoy core wars on your A8 with this package. It is compatible with the unofficial ICWS-88 standard. There are also some features from the ICWS-94. Extract the TAR file and see the man pages for detailed information.

### 10.2.3 Toys - 4 small programs

'Blink' offers a blinking cursor with adjustable blinking speed that is RESET proof.

'HQ9' is an A8 version of Biffle's 'HQ9+'. Type 'HQ9 /H' for information.

'Life' brings Conway's Game of Life to the A8. It starts with a usage screen.

'Sieve' is an A8 version of 'The Sieve of Eratosthenes' that derives all prime numbers from up to a 16-bit value. Type 'SIEVE /H' for instructions.

## 10.3 Contributed by FujiDude

### 10.3.1 AAC - ASCII<->ATASCII converter

Conversion tool written in Action! with sources and documentation.

## 10.4 Contributed by mono

These add-ons all come with man pages providing information and instructions.

### 10.4.1 Worm - a screen saver

A spermatozoid screen saver that allows to set the delay in seconds. Useful when running a CRT monitor or CRT TV. Works on VBXE as well.

### 10.4.2 Drum Machine - convert your Atari into a drum kit!

DRUM Machine for SDX that plays NEO Tracker or WAV samples. Supported devices are POKEY, a second POKEY (\$D210 by default), COVOX (\$D600 by default), and SimCOVOX (\$D300 by default).

### 10.4.3 CMC Play - Chaos Music Composer player

CMC plays CMC/CM3/CMR/DMC music. Second POKEY for stereo is detected at \$D210.

### 10.4.4 FC Play - Future Composer player.

Plays FC music files. Second POKEY for stereo is detected at \$D210.

### 10.4.5 MPT Play - Music Protracker player.

Plays MPT/MD1/MD2 music files. Second POKEY for stereo is detected at \$D210.

### 10.4.6 RMT Play - Raster Music Tracker player.

Plays RMT music files. Second POKEY for stereo is detected at \$D210.

### 10.4.7 ST Play - Sound Tracker player.

Plays MUZ/ST7/ST8/ST music files. Second POKEY for stereo is detected at \$D210.

**10.4.8 TMC Play - Theta Music Composer player.**

Plays TMC music files. Second POKEY for stereo is detected at \$D210.

**10.5 Diamond Desktop**

Version 2.0 ROM patched for SDX compatibility.



## Appendices

### A DOS limitations

#### The maximum capabilities list

A.	Number of drives or partitions:	15
B.	Logical sector size:	512 bytes
C.	Number of sectors per medium:	65535
D.	Medium size (B*C):	33553920 bytes (~32 MiB)
E.	Total capacity (A*D):	503308800 bytes (~480 MiB)
F.	Directory size:	32768 bytes (32 KiB)
G.	Number of directories:	unlimited
H.	Number of entries per directory:	1423
I.	Number of files per medium (G*H):	unlimited
J.	File size:	16777216 bytes (16 MiB)
K.	Number of files open at a time:	16
L.	Path length:	64 characters (incl. terminating EOL)
M.	Extended memory:	max. 1024 KiB with Port B type, max. 4032 KiB with Axlon type



## B Error Messages

### B.1 Description of all error messages

The following is a list of error codes and messages that may occur while using SDX. Some of the more common error codes are displayed in message form (as indicated by quotes) with most SDX commands. Other programs may display error messages or just simply an error code (in either decimal or HEX (\$) form). Following each error code and message (if applicable) is a description of what probably caused the error. All error codes less than 128 (\$80) are application errors (BASIC, ACTION!, etc.) and are not produced by SDX.

The descriptions here are meant to cover the most common error conditions. It is possible to get some of these error codes or messages under different circumstances. If this happens, please remember to check also the error codes of the hardware used.

#### **128    \$80    'User break abort'**

The BREAK key was pressed when the computer was waiting for input or printing to the screen. BREAK does not interrupt disk I/O in SDX, but most programs will terminate after disk I/O is completed if the BREAK key has been pressed.

#### **129    \$81    'File already open'**

Attempted to open a file for output that is already open. Can occur if you try to COPY a file on top of itself. For example, 'COPY MYFILE'. This error will occur since the default destination is '\*.\*'. No prior versions of SD made this type of check, so it was easy to inadvertently lose files using COPY.

This error will also occur when opening a file through the CIO, if the IOCB had not been closed properly. This is a problem in some Atari programs (most notably the ACTION! cartridge). The command processor makes sure all IOCBs are closed before entry, so this error usually occurs within programs.

#### **130    \$82    'Nonexistent device'**

The device specifier that was provided does not exist. Valid device specifiers for the SDX command processor are DSK:, CAR:, CLK:, PRN:, CON: and COM:. Through the CIO the valid devices are D:, E:, C:, S:, K:, R: and P:. Of course devices may be added, but these are the standard devices.

#### **131    \$83    'File is write-only'**

Attempt to read from a file that was open for write only (mode 8 or 9). This error indicates a programming error.

#### **132    \$84    'No device handler installed' (Bad CIO command)**

Attempt to call the CIO with an invalid function code. Note that all function codes above 13 are considered XIO calls and therefore will not return this error. They return 'No function in device handler' instead. This error indicates a programming error like an attempt to access a 'kernel' device (e. g. COM:) with no handler installed.

#### **133    \$85    (File not open)**

Attempt to perform a read or write (or note/point XIO operation) on a file that has not yet been opened. Indicates a programming error.

**134     \$86     'Bad file handle'**

Call of the CIO with an invalid IOCB number in the X register. You must multiply the IOCB number you wish to use by 16. Indicates a programming error.

**135     \$87     'File is read-only'**

Attempt to write to a file that was open for read only (mode 4). Indicates a programming error.

**136     \$88     (End Of File)**

Not really an error but an indication of the end-of-file. This status may only be returned by an input function through the CIO. SDX kernel calls return EOF status differently.

**137     \$89     'Truncated record'**

Not really an error but an indication that the record attempted to read was longer than the read-in buffer given. This status may only be returned by an input function through the CIO. The SDX kernel calls return this status differently.

**138     \$8A     'Device does not respond'**

Attempt to access a drive that was either non-existent, turned off or disconnected. Also, your drives may have been SWAPped (see SWAP command). Check your SIO cables, power cords and Multi I/O menu (if applicable).

**139     \$8B     'Device NAK'**

This error occurs when parameters for the drive I/O operation (like read/write sector) are out of range or the SIO command is unknown. The following conditions may apply:

- Disk drive door is open.
- MIO is configured for a hard drive but none is online at that drive number.
- A bad sector found and the drive takes a long time to return any response.
- Try to read an illegal ATR image. This can get the SIO out of sync and result in a 'Drive NAK' error.

**140     \$8C     'SIO framing error'**

This error indicates that drive and computer are not communicating properly. If this error consistently happens, then probably drive or computer needs service (Unlikely but possible that a serial bus overrun error can occur if a bad sector is found).

**141     \$8D     'Cursor out of range'****142     \$8E     'SIO overrun'**

This error indicates that drive and computer are not communicating properly. If this error consistently happens, then probably drive or computer needs service (Unlikely but possible that a serial bus overrun error can occur if a bad sector is found).

**143     \$8F     'SIO checksum error'**

This error indicates that your drive and computer are not communicating properly. If this error consistently happens, then you probably need your drive or computer serviced (Unlikely but possible that a serial checksum error can occur if a bad sector is found).



**144 \$90 'Write protected or bad sector'**

If you are reading from a medium, this error indicates that a sector is bad. If you are writing to it, you either have the medium write protected or the sector SD is trying to write does not exist (either because of a configuration problem or the sector has a bad header). Note that when you 'Lock' a drive through the Multi I/O menu, you will get a 'Drive NAK' error instead.

**146 \$92 'No function in device handler'**

You attempted to perform a command on a device that does not support that command. For example you cannot RENAME a file on the CAR: device or perform a directory listing of PRN:. On the CIO level, this indicates that the XIO function you attempted does not exist on the device specified.

**148 \$94 'Unknown filesystem'**

SDX can not recognize the DOS format of the disk attempted to access. If the disk is AtariDOS 2 format, then make sure that the ATARIDOS.SYS driver is installed in the system. It is installed by default if there is no CONFIG.SYS on drive 1. If there is a CONFIG.SYS on drive 1, make sure that the line 'DEVICE ATARIDOS' is included. See also note with err 139.

**150 \$96 'Path not found'**

The specified directory path does not exist. Recheck the specified pathname. Perform a directory command on each directory of the path to make sure that they all exist.

**151 \$97 'File exists'**

Attempts to overwrite a file that is protected, replace a directory with a file, or replace a file with a directory. Or you tried to rename a file with an already existing filename.

**152 \$98 'Not binary file'**

Attempt to LOAD or run a file that is not a binary load file. There are several scenarios in which this error can occur.

- The file does not start with a valid binary file header of \$FFFA or \$FFFF. (The file is a BASIC program, text file, database, etc.)
- You attempted to run a relocatable SDX command file with the X command. The X command only loads standard Atari binary load files.
- The end of the file you attempted to load has been corrupted. This is generally caused by incompatible communications software when the file was either uploaded or downloaded from a bulletin board.

**154 \$9A 'Symbol not defined'**

The SDX loader could not load a program because it accessed a symbol that has not been defined. This indicates that first the appropriate driver for the command needs to be loaded. For example, the TD.COM command needs either the DRIVER.SYS of the used realtime clock or the JIFFY.SYS driver to be installed. These drivers define a symbol called I\_GETTD which is referenced by TD to get the current time/date.

**156 \$9C 'Bad Parameter'**

An invalid parameter has been given to a command. Refer to the appropriate command description in this reference guide for command syntax and usage.

**158 \$9E 'Out of memory'**

Attempt to load or run a SDX command that will not fit in memory. Make sure that there are no programs 'held' in memory (see the LOAD command). If you are still out of memory, then reboot with fewer drivers and try again. The only case where not enough memory is available is when attempting to ARCHive files on a stock Atari 800 computer.

**161 \$A1 'Too many channels open'**

SDX supports up to 16 open files, but each driver has its own limitation. The DSK: driver allows you to specify the maximum number of channels open to it (the default is 5 which should be enough for any application). The CAR: driver only has 1 channel which means that you may not copy files from CAR: with the COPY command. (COPY uses two channels on the source device and one on the destination. This is because COPY opens one channel to the directory and another to the file to be copied.) To overcome this limitation, you may TYPE the file on CAR: and redirect output to a file on disk (e. g. TYPE CAR:COMMAND.COM >>NEW.COM).

If error 161 occurs, increase the number of file buffers. This is done in the CONFIG.SYS file with the SPARTA.SYS driver as described in chapter 8. Just increase your 'nfiles' value by one or more. Increasing 'nbufs' will speed up disk access for additional open files but is not required.

**162 \$A2 'Disk full'**

The disk is full. SDX directories handle up to 1423 files so it is probably a full disk. If you were copying files to the disk that became full, the last copied file is removed from it.

**163 \$A3 'Illegal wildcard in name'**

You may not use wildcards when modifying or creating a file or creating a subdirectory. Wildcards are allowed when opening a file for input or in a directory path.

**165 \$A5 'Bad filename'**

The entered filename has a bad character in it. The two most common places for this error to occur are entering a bad character in a directory path or using a bad delimiter in the RENAME command.

**166 \$A6 'Range error'**

In a file operation this means: while reading, an attempt to read data or seek past the end of the file; while writing, the file exceeded its size limit (the limits are: 16 MiB for a regular file and 32 KiB for a directory). Generally: a parameter for the operation is beyond the allowed limit.

**167 \$A7 'Directory not empty'**

Attempt to delete a directory that still contains files or subdirectories. First ERASE all files and delete all subdirectories including those that are hidden.

**Note:** A file opened for write or update but not closed properly (usually due to a system reset or power loss while open) will leave a 'phantom' entry in the directory. The 'phantom' entry prevents deletion. Use 'CleanUp X' from SDXTK to remove this entry to allow the directory to be deleted (→ Appendix E).

**169 \$A9 'Directory full'**

A new file cannot be created, because there is no space left in the directory to store its name. A directory may contain maximum 1423 entries for user files and directories.

**170    \$AA    'File not found'**

The file you tried to access does not exist. This error will also occur if you attempt to rename or erase a protected file.

**176    \$B0    'Access denied'**

The first block\_io function called for a disk was not function 4 (bio\_rdsys), the disk drive number specified equals 0 or is greater than 15, or an invalid function number was specified. Appears also when a physical sector size is bigger than 512 bytes is detected.

**179    \$B3    'Memory conflict'**

An attempt to load a program, which overlaps the DOS kernel or the I/O library area. It often indicates, that the program has to be executed using X.COM.

**181    \$B5    'File system corrupt'**

The DOS cannot do the requested operation, because the file system structure on the disk is damaged.

**182    \$B6    'Path too long'**

The length of the pathname created by the program is greater than the allowed limit. Paths are currently limited to 64 characters.

**183    \$B7    'Environment full'**

The default space for environment variables (which is only 256 bytes) is completely filled up. Release some space by deleting some variables or load ENV.SYS.

**184    \$B8    'Wrong architecture'**

Signals an attempted execution i. e. of a 65C816 program on 6502.

**Notes:** Error codes not covered by a description generated by SDX will be displayed with the error code number followed by the message, e. g. '180 System error'.

The MSC IDE controller generates error codes 176-183 with a different meaning than those generated by SDX.

## B.2 Error Message Summary

128	\$80	User break abort
129	\$81	File already open
130	\$82	Nonexistent device
131	\$83	File is write-only
132	\$84	No device handler installed (Bad CIO command)
133	\$85	(File not open)
134	\$86	Bad file handle
135	\$87	File is read-only
136	\$88	(End Of File)
137	\$89	Truncated record
138	\$8A	Device does not respond
139	\$8B	Device NAK
140	\$8C	SIO framing error
141	\$8D	Cursor out of range
142	\$8E	SIO overrun
143	\$8F	SIO checksum error
144	\$90	Write protected or bad sector
146	\$92	No function in device handler
148	\$94	Unknown filesystem
150	\$96	Path not found
151	\$97	File exists
152	\$98	Not binary file
154	\$9A	Symbol not defined
156	\$9C	Bad Parameter
158	\$9E	Out of memory
161	\$A1	Too many channels open
162	\$A2	Disk full
163	\$A3	Illegal wildcard in name
165	\$A5	Bad filename
166	\$A6	Range error
167	\$A7	Directory not empty
169	\$A9	Directory full
170	\$AA	File not found
176	\$B0	Access denied
179	\$B3	Memory conflict
181	\$B5	File system corrupt
182	\$B6	Path too long
183	\$B7	Environment full
184	\$B8	Wrong architecture

**Notes:** Error codes not covered by a description generated by SDX will be displayed with the error code number followed by the message, e. g. '180 System error'.

The MSC IDE controller generates error codes 176-183 with a different meaning than those generated by SDX.

## C Command Summary - Alphabetical

### Quick Reference List

Quick reference for syntax and usage. For more details concerning operation please refer to chapter 4 (commands), 5 (batch processing), and 8 (system configuration).

#### **-fname [param1, param2 , ... , param9]**

Executes the specified batch file, optionally passing parameters. The extension is assumed to be '.BAT' if none is specified.

#### **APPEND d: | pathname | device**

Append the given drive and/or path at the end of the \$PATH variable.

#### **ARC command[option] [d:][path]arcfname[.ext] [d:][path][filelist]**

Create and maintain file archives. Commands: A, M, U, F, D, X, E, P, L, V. Options: B, S, W, N, H, G.

#### **ATR [+|-][A|H|P] [d:][path]fname[.ext]**

Set/clear file attributes in the directory. Replaces the Protect and Unprotect functions from older SD versions. Alias - ATTRIB.

#### **BASIC [/I|N] [d:][path][frame] [parameters]**

Enter the internal BASIC in a XL or XE computer (except 1200XL).

#### **BLOAD [d:][path]fname[.ext] [[\$]index:][\$] address**

Load a file into the given memory area starting at the given address.

#### **BOOT [d:][path]fname[.ext]**

Tell a SDFS formatted medium to boot a particular program at start up.

#### **CAR [/F|N|I|L S|A|N|I|mode] [d:][path][fname] [parameters]**

Enter the cartridge plugged into the top of the SDX cartridge. If a filename is specified, then that binary file is loaded and handled according to the given parameters.

#### **CHDIR [d:][path]**

Change the current (working) directory on the specified drive, or displays the current directory path if no path is given. Alias - CD, CWD.

#### **CHKDSK [d:] [/X]**

Show volume, free/total disk space and sector size of the medium in the selected drive.

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

Change the time/date stamp on all files matching the given filespec to the current time and date.

#### **CHVOL [d:]volname**

Change the volume name on the specified drive.

#### **CLR**

Delete the environment variables created by the system that are no longer used.

**CLS [/F]**

Simply clear the screen. Especially for batch files it is very useful.

**COLD [/C|N]**

Reboot the system (by doing a jump through \$E477).

**COMMAND (The Command Processor)**

Enter commands and run other programs. It is not entered as a command itself but is automatically invoked when you enter DOS.

**COMP [d:][path]fname1.ext [d:][path]fname2.ext [offset1 [offset2]]**

Compare two files.

**CON 40|64|80**

Enables and disables the 64- (on CAR:) and 80-column (on SDXTK) console text modes.

**COPY [/B|C|D|I|K|M|N|Q|R|S|V] [+][A|H|P] [d:][path][fname].[ext] [d:][path][fname].[ext]/[A]**

Copy file(s) to another drive and, optionally, give the copy a different name.

**DATE [/T|dd[-mm][-yy]|mm[-dd][-yy]]**

Displays the current date and allows you to set the date. DAYTIME settings apply here.

**DELTREE [/YV] [d:] path**

Delete subdirectory trees recursively.

**DEV**

Display the list of available/installed kernel devices.

**DEVICE driver**

Used in CONFIG.SYS to load installable drivers.

**DF [/A]**

Display summary information about free space on all disks.

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

Displays a long (short) formatted directory with byte size, date and time (DOS 2 type).

**DUMP [d:][path]fname.ext [[\$]start] [[\$]len] [/A]**

Displays a file in HEX and ATASCII form.

**ECHO ON|OFF**

Enable or disable the 'echo' in the Command Processor.

**ECHO string**

Displays a message during the execution of the CONFIG.SYS.

**ED [d:][path][filename.ext]**

Enable text editor.

**ERASE [d:][path]fname[.ext]**

Deletes the file in the specified directory on the designated drive, or deletes the file from the default drive if no drive is specified. If no path is specified, the file is deleted from the current directory. Alias - DEL & DELETE.

**EXIT [n]**

Causes the immediate termination of a processing batch file. Optional exit code to be taken by the system as error code.

**FIND [d:]device]fname[.ext]**

Searches all directories on all drives/file oriented devices for files matching the given filespec.

**FMT [/S|J] [<<]fname[.ext] [ncol]**

Simple text formatter that reads the input line by line and applies the specified options.

**FOR ... NEXT**

Use of counted loops in batch files.

**FORMAT [/Q dn: [volname]]**

Initializes any medium in SD or AtariDOS 2 format.

**GOSUB**

Calls ordinary GOTO-labels, provided the 'labeled' command sequence is ended with 'RETURN'.

**GOTO**

Allows to make a jump within a batch file.

**IF [NOT] - ELSE - FI**

Use of conditional expressions in batch files.

**INKEY**

Stops batch processing and waits for a key press. Conditions may apply.

**KEY ON|OFF**

Installs a 32 character keyboard buffer and links an 'internal' KEY command into your system (for turning the buffer on/off).

**LESS [/C] [path]fname[.ext] [>>][path]fname[.ext]]**

Text viewer with smart options. Converts MS-DOS and UNIX text files to Atari format.

**LOAD [/X|L S|A|N|I|mode] [d:][path][fname][.ext] [parameters]**

Loads a file (no run). If no filename is used, all files previously loaded are removed from memory.

**MAN [fname] [/P|?]**

Starts the documentation viewer.

**MAP [unit] [SIO|OS|NORMAL|OFF] [d:] or MAP [d:]fname[.ext]**

SIO.SYS control.

**MDUMP [[*\$*]index:][*\$*]address [[*\$*]len] /R**

Display memory contents in hex and ATASCII. Can access 65C816 high ram.

**MEM [/X]**

Displays the current low memory limits of your system, the available windowed RAM, the overall extended memory and its type.

**MENU or the <\*> key**

Allows to perform COPY, ERASE, RENAME, etc. commands on all selected files. It is similar to other SD menu programs, but provides many new features.

**MERGE [d:][path]fname[.ext]**

Allows merging of text files with CONFIG.SYS; enables a modular built CONFIG.SYS.

**MKDIR [d:]path**

Creates a subdirectory. Alias - MD & CREDIR.

**MORE [<<]fname[.ext]**

Displays the contents of the given text file.

**PATH [path\_string]**

Causes specified directories to be searched for commands before searching the current directory.

**PAUSE [n]**

Suspends system processing and displays the message 'Press RETURN to continue'. Optionally accepts a number of seconds to wait, ranged from 0 to 65535.

**PEEK symbol | [[*\$*]index:][*\$*]address or PEEK symbol**

Examines a memory location, performs a HEX conversion, or converts the given symbol to the associated address and memory index. Can access 65C816 high ram.

**POKE symbol | [[*\$*]index:][*\$*]location [*\$*]value**

Changes the content of a memory location. Can access 65C816 high ram.

**PROC**

Starts a procedure in a batch file.

**PROMPT [prompt\_string]**

Change the system prompt.

**PWD**

Outputs a list of current working directories.

**RENAME [d:][path]fname[.ext] fname[.ext]**

Changes the name of one or more files. Alias - REN.

**RENDIR [d:][path]dir\_name\_old dir\_name\_new**

This command allows you to change the name of a directory.

**RETURN**

Ends a process in a batch file.



**RMDIR [d:]path**

Deletes an empty subdirectory from the specified drive. Alias - RD & DELDIR.

**RS232**

Loads the RS232 handler from a P:R: Connection or the Atari 850 interface.

**SAVE [d:][path]fname[.ext] [\$]address [\$]address**

Saves binary data from memory to disk.

**SET [var[=env\_string]]**

Display the values of all environment variables or set an environment variable.

**SETPATHS [d:][path] | fname[.ext]**

Invoked from a batch file the predefined paths are set on the addressed drives. Alternatively the required path can be specified directly as a command line argument.

**SIOSET [d: [type [usindex]]] or****SIOSET REFRESH [0|128] or****SIOSET NMI [index] or****SIOSET WAITACK [[\$]index]**

SIO.SYS serial speed control.

**SORTDIR [d:][path] [/N|T|S|D|X]**

To sort filenames in directories by name, type, date or size.

**SWAP [d d]**

Swap (re-map) your drive configuration or display the current drive map list.

**TD [ON|OFF]**

Turns on and off a time/date display line on top of your screen.

**TIME [/T][hh[:mm]][:ss]**

Displays the current time and allows to set the time.

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

Displays the contents of a specified file.

**UNERASE [d:][path]fname[.ext]**

Restores files previously erased (if possible).

**USE OSRAM|BANKED|NONE**

Defines in CONFIG.SYS what secondary RAM area to use.

**VER**

Displays the current version number and date of the cartridge.

**VERIFY [ON|OFF]**

Turns write verify on or off.

**X [/C|L S|A|N|I|mode] [d:][path]fname[.ext] [parameters]**

Execute a program which requires no cartridge being installed.



## D Command Summary - By Function

Quick reference for syntax and usage. For more details concerning operation please refer to chapter 4 (commands), 5 (batch processing), and 8 (system configuration).

### D.1 Batch Files

---

#### **-fname [param1, param2 ,..., param9]**

Executes the specified batch file, optionally passing parameters. The extension is assumed to be '.BAT' if none is specified.

#### **CLR**

Deletes the environment variables created by the system that are no longer used.

#### **CLS [/F]**

This command simply clears the screen. Especially for batch files it is very useful.

#### **ECHO ON|OFF**

Enable or disable the 'echo' in the Command Processor.

#### **EXIT [n]**

Causes the immediate termination of a processing batch file. Optional exit code to be taken by the system as error code.

#### **FOR ... NEXT**

Use of counted loops in batch files.

#### **GOSUB**

Calls ordinary GOTO-labels, provided the 'labeled' command sequence is ended with 'RETURN'.

#### **GOTO**

Allows to make a jump within a batch file.

#### **IF [NOT] - ELSE - FI**

Use of conditional expressions in batch files.

#### **INKEY**

Stops batch processing and waits for a key press. Conditions may apply.

#### **PAUSE [n]**

Suspends system processing and displays the message 'Press RETURN to continue'. Optionally accepts a number of seconds to wait, ranged from 0 to 65535.

#### **PROC**

Starts a procedure in a batch file.

#### **RETURN**

Ends a process in a batch file.

**SETPATHS [d:][path] | fname[.ext]**

Invoked from a batch file the predefined paths are set on the addressed drives.

**D.2 Configuration Commands**

---

**DEVICE driver**

Used in CONFIG.SYS to load installable drivers.

**ECHO string**

Display of a message during the execution of the CONFIG.SYS.

**MERGE [d:][path]fname[.ext]**

Allows merging of text files with CONFIG.SYS; enables a modular built CONFIG.SYS.

**SET var=env\_string**

Set an environment variable in CONFIG.SYS or AUTOEXEC.BAT.

**USE OSRAM|BANKED|NONE**

Defines in CONFIG.SYS what secondary RAM area to use.

**D.3 Directory Commands**

---

**CHDIR [d:][path]**

Changes the current (working) directory on the specified drive, or displays the current directory path if no path is given. Alias - CD & CWD.

**DELTREE [/YV] [d:] path**

Delete subdirectory trees recursively. Optionally, you can watch the directories with their containing files being deleted.

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

Displays a long (short) formatted directory with byte size, date and time (DOS 2 type).

**MENU or the <\*> key**

Allows to perform COPY, ERASE, RENAME, etc. commands on all selected files. It is similar to other SD menu programs, but provides many new features.

**MKDIR [d:]path**

Creates a subdirectory. Alias - MD & CREDIR.

**RENDIR [d:][path]dir\_name\_old dir\_name\_new**

This command allows you to change the name of a directory.

**RMDIR [d:]path**

Deletes an empty subdirectory from the specified drive. Alias - RD & DELDIR.

**SETPATHS [d:][path] | fname[.ext]**

Invoked from a batch file the predefined paths are set on the addressed drives. Alternatively the required path can be specified directly as a command line argument.

**SORTDIR [d:][path] [/N|T|S|D|X]**

To sort filenames in directories by name, type, date or size.

**D.4 Disk Maintenance Commands**

---

**BOOT [d:][path]fname.ext**

Tells a SDFS formatted medium to boot a particular program at start up.

**CHKDSK [d:] [/X]**

Shows volume, free/total disk space and sector size of the medium in the selected drive.

**CHVOL [d:]volname**

This command changes the volume name on the specified drive.

**DF [/A]**

Display summary information about free space on all disks.

**FORMAT [/Q dn: [volname]]**

Initializes any medium in SD or AtariDOS 2 format.

**MAP [unit] [SIO|OS|NORMAL|OFF] [d:] or MAP [d:]fname.ext**

SIO.SYS control.

**PWD**

Outputs a list of current working directories.

**SIOSET [d: [type [usindex]]] or**

**SIOSET REFRESH [0|128] or**

**SIOSET NMI [index] or**

**SIOSET WAITACK [[\$]index]**

SIO.SYS serial speed control.

**VERIFY [ON|OFF]**

Turns write verify on or off.

**D.5 File Maintenance Commands**

---

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

Sets/clears file attributes in the directory. Replaces the Protect and Unprotect functions from older SD versions. Alias - ATTRIB.

**BLOAD [d:][path]fname.ext [[\$]index:][\$] address**

Loads a file into the given memory area starting at the given address.

**COMP [d:][path]fname1.ext [d:][path]fname2.ext [offset1 [offset2]]**

Compares the given files.

**COPY** [/B|C|D|I|K|M|N|Q|R|S|V] [+][A|H|P] [d:][path][fname].[ext] [d:][path][fname].[ext]/A

Copies one or more files to another drive and optionally gives the copy a different name if you specify it in the COPY command.

**ERASE** [d:][path]fname.[ext]

Deletes the file in the specified directory on the designated drive or deletes the file from the default drive if no drive is specified. If no path is specified, the file is deleted from the current directory. Alias - DEL & DELETE.

**MENU** or the <\*> key

Allows to perform COPY, ERASE, RENAME, etc. commands on all selected files. It is similar to other SD menu programs, but provides many new features.

**RENAME** [d:][path]fname.[ext] fname.[ext]

Changes the name of one or more files. Alias - REN.

**UNERASE** [d:][path]fname.[ext]

Restores files previously erased (if possible).

## D.6 Running Programs

---

**BASIC** [/I|N] [d:][path][fname] [parameters]

Enters the *internal* BASIC in a XL or XE computer (except 1200XL).

**CAR** [/N|I|L S|A|N|I|mode] [d:][path][fname] [parameters]

Enters the cartridge plugged into the top of the SDX cartridge. If a filename is specified, then that binary file is loaded and handled according to the given parameters.

**X** [/C|L S|A|N|I|mode] [d:][path]fname.[ext] [parameters]

Execute a program which requires that no cartridges are installed using specific loading modes.

## D.7 Command Processor Options

---

**COLD** [/C|N]

Reboots the system (by doing a jump through \$E477).

**COMMAND** (The Command Processor)

Allows to enter commands and run other programs. It is not entered as a command itself but is automatically invoked when you enter DOS.

**KEY ON|OFF**

Installs a 32 character keyboard buffer and links an 'internal' KEY command into your system (for turning the buffer on/off).

**PATH** [path\_string]

Causes specified directories to be searched for commands before searching the current directory.

**PROMPT [prompt\_string]**

Change the system prompt.

**RS232**

Loads the RS232 handler from a P:R: Connection or the Atari 850 interface.

**SET [var[=env\_string]]**

To display the values of all environment variables and optionally sets an environment variable to a specified value..

**SWAP [d,d]**

Swap (re-map) your drive configuration or display the current drive map list.

**D.8 Time - Date Support**

---

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

Changes the time/date stamp on all files matching the given filespec to the current time and date.

**DATE [/T[dd[-mm]][-yy][mm[-dd]][-yy]]**

Displays the current date and allows you to set the date. DAYTIME settings apply here.

**TD ON|OFF**

Turns a time/date display line on top of your screen on and off.

**TIME [/T[hh[:mm]][:ss]]**

Displays the current time and allows to set the time.

**D.9 Utilities And Programming Aids**

---

**APPEND d: | pathname | device**

Append the given drive and/or path at the end of the \$PATH variable.

**ARC command[option] [d:][path]arcfname[.ext] [d:][path]{filelist}**

Creates and maintains file archives. Commands: A, M, U, F, D, X, E, P, L, V. Options: B, S, W, N, H, G.

**CON 40|64|80**

Enables and disables the 64- (on CAR:) and 80-column (on SDXTK) console mode.

**DEV**

Display the list of available/installed kernel devices.

**DUMP [d:][path]fname[.ext] [[[\$]start] [[[\$]len] [/A]**

Displays a file in HEX and ATASCII form.

**ED [d:][path][filename.ext]**

Enable text editor.

**FIND [d:]device]fname[.ext]**

Searches all directories on all drives/file oriented devices for files matching the given filespec.

**FMT [/S|J] [<<]fname[.ext] [ncol]**

A simple text formatter that reads the input line by line and applies the following to specified options.

**LESS [/C] [path]fname[.ext] [>>[path]fname[.ext]]**

Text viewer with smart options. Converts MS-DOS and UNIX text files to Atari format.

**LOAD [/X|L S|A|N|I][mode] [d:][path][fname][.ext] [parameters]**

Loads a file (no run). If no filename is used, all files previously loaded are removed from memory. This is useful for keeping commonly used commands resident in memory, thereby eliminating the need for these commands to load from disk. Switches allow special conditions.

**MAN [fname] [/P|?]**

Starts the documentation viewer.

**MDUMP [[\$]index:][\$]address [[\$]len] /R**

Display memory contents in hex and ATASCII. Can access 65C816 high ram.

**MEM [/X]**

Displays the current low memory limits of your system, the available windowed RAM, the overall extended memory and its type.

**MORE [<<]fname[.ext]**

Displays the contents of the given text file.

**PEEK symbol | [[\$]index:][\$]address or PEEK symbol**

Examines a memory location, performs a HEX conversion, or converts the given symbol to the associated address and memory index. Can access 65C816 high ram.

**POKE symbol | [[\$]index:][\$]location [\$]value**

Changes the content of a memory location. Can access 65C816 high ram.

**SAVE [d:][path]fname[.ext] [\$]address [\$]address**

Saves binary data from memory to disk.

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

Displays the content of a specified file.

**VER**

Displays the current version number and date of the cartridge.



## E Miscellaneous Notes

### E.1 Using Turbo-BASIC XL with SDX

The well-known Turbo-BASIC XL (TBXL) uses the RAM under the OS ROM in XL/XE computers. Therefore it is not compatible with SD 2.x and 3.x. However, with the proper hardware and configuration TBXL will work well with SDX.

#### E.1.1 Hardware Configuration

A XL/XE computer with more than 64 KiB is needed to use TBXL with SDX. The 400/800 computers do not have RAM under the OS. XL/XE computers with 64 KiB memory or less are also unsuitable, since SDX needs to acquire the RAM under the OS on these machines (USE OSRAM). This conflicts with TBXL using the same areas.

#### E.1.2 System Configuration

One bank of extended memory (USE BANKED) is needed to run TBXL. If you have more than 128 KiB of RAM in your computer, this will occur by default. If you have just 128 KiB however, it is required that you boot with a custom CONFIG.SYS file from a SDFS formatted medium to be able to use TBXL. See chapter 8 about the 'config selector' for a possible alternative.

The first line of any CONFIG.SYS while using TBXL must be

```
USE BANKED
```

The rest of the CONFIG.SYS file is up to you. Do not forget to include DEVICE SPARTA and DEVICE SIO in this file. The following is an example of a CONFIG.SYS that will work with any XL/XE computer with more than 64 KiB and TBXL:

```
USE BANKED
DEVICE SPARTA
DEVICE SIO
DEVICE JIFFY (or a driver for your RTC)
DEVICE RAMDISK 0,128
```

You can create the CONFIG.SYS file using the ED command. This is just an example. You may change it as you see fit as long as the first line is USE BANKED. Finally, don't forget to use X.COM when loading TBXL, its compiler or runtime, e. g. 'X COMPILER'.

**Note:** The file system drivers have been modified to make the BLOAD command of Turbo-BASIC XL work with SDX.

## **E.2 Using AUTORUN.SYS files**

---

SDX, when booted, will not automatically load and run a file named AUTORUN.SYS. With batch files and the relocatable nature of the SDX command processor the need of such a file is eliminated. There are three major types of AUTORUN.SYS you are likely to encounter. See their descriptions and the best way to handle them.

### **E.2.1 Applications**

Many programs are named AUTORUN.SYS simply to have them load and run when the computer is booted. These files will usually be fairly long and take control of the computer when run. To use these, simply rename them to a relevant name and type that name from the command line. It may be necessary to use the X command to have the program perform correctly. It is not necessary to rename the program, but it is much more convenient to have the name of the file reflect its function and to be able to store several of these formerly AUTORUN.SYS files on one disk.

### **E.2.2 Handlers**

Many AUTORUN.SYS files install device handlers into the CIO's handler table. Among these are RS232 and other modem handlers and custom handlers, such as the G: device from ANALOG Computing (issue #35). These are usually short files and return control to the command processor or the language cartridge shortly after loading. These, too, can be renamed to some other name (such as RS232.COM or G.COM) and run from the command line.

### **E.2.3 BASIC Program Loaders**

The third common type of AUTORUN.SYS file is a machine language program that loads and runs a BASIC program from disk. These are usually found on magazine disks. To use one of these, you may simply rename it to something like MENU.COM and type

```
BASIC /N MENU
```

### **E.2.4 Using Batch Files**

Any of these programs or a group of these programs can be run automatically by using batch files. Simply create a text file containing a list of the programs you wish to run and name it AUTOEXEC.BAT. When the computer is booted with this AUTOEXEC.BAT, the commands in the list will automatically be executed. For more information on batch files, please refer to chapter 8.

## **E.3 Using ACTION! with SDX**

---

The KEY buffer works very well with ACTION!. It has been advanced to be compatible with more programs than it used to be with earlier versions of SD.

ACTION! has some characteristics the user needs to adapt to. This is also true with SDX. Therefore the default loading mode with CAR since SDX 4.47 is set to the AtariDOS compatible mode 64. Only the INIT and RUN vectors are to be used then. ACTION! compiled binaries, which set INITs but do not properly set the RUN vector, will now be executed as expected.

ACTION! comes with its own screen accelerator. Disable QUICKED.SYS if present in your CONFIG.SYS to avoid conflicts.

## E.4 Using BASIC XL/XE with SDX

---

BASIC XE uses the same OSRAM area that the SPARTA.SYS driver uses for buffers if the 'OSRAM' parameter is given. This means that you cannot use 'DEVICE SPARTA OSRAM' in your CONFIG.SYS file when using BASIC XE. This only applies when 'USE OSRAM' is the first line in your CONFIG.SYS file, since the 'OSRAM' parameter for SPARTA.SYS is ignored otherwise.

This also means that if you are using a 64 KiB or 128 KiB XL/XE computer you must use a custom CONFIG.SYS file to run BASIC XE. To create one, see chapter 8. You can also refer to the example in 'Using Turbo-BASIC XL'. You may modify it to your needs as long as the first two lines are

```
USE OSRAM
DEVICE SPARTA
```

BASIC XL/XE come with an own screen accelerator. Disable QUICKED.SYS if present in your CONFIG.SYS to avoid conflicts.

## E.5 Using BASIC XE Extensions

---

The disk-based extensions for BASIC XE provide many useful tools for the programmer. They can also present a few problems for users of SDX. Fortunately, these problems can be easily avoided.

### E.5.1 Loading the Extensions

Due to the shortage of free RAM, the DOS now uses the area at \$D800-\$DFFF to keep its internal structures while in USE OSRAM mode. BASIC XE Extensions load into the same place, therefore SDX must be configured in BANKED mode and the computer has to have more than 128 KiB of RAM.

### E.5.2 Other Issues

Once loaded, the extensions will still be there, whether you use internal BASIC or the X command to run programs. This can cause conflicts with the programs and will almost certainly cause problems while attempting to use BASIC XE again. It is recommended to perform a COLD start to wipe out the extensions before running other programs.

When using the memory save function of CAR, it will be different when using the extensions.<sup>99</sup>

## E.6 Using MAC/65 and DDT with SDX

---

MAC/65 works well with SDX, but DDT, the debugger in the MAC/65 cartridge, will not operate properly with the key buffer active (KEY ON). The simple solution is to either do a KEY OFF before entering the cartridge or not installing the key buffer at all.

Cold initialization of MAC/65 takes place just before entering the cartridge for the first time (i. e. when the first CAR command is issued). When you exit to DOS, MAC/65 state

---

<sup>99</sup> See Chapter 4 - CAR command.

is saved in the '.SAV' file. Later, on the cartridge re-entry, the initialization is skipped and the cartridge state is restored. If you want to force another cold initialization, enter CAR with the '/I' switch.

## **E.7 Using AtariWriter Plus with SDX**

---

With a stock 130XE or 800XL computer, using AtariWriter Plus is straightforward. Simply insert the AtariWriter Plus disk into D1: and type 'D1:**X AP.OBJ**'. If you have more than 128 KiB of RAM in your computer, the procedure is a bit more complex. You will need to prepare a boot floppy disk for AtariWriter Plus in SD format and create a CONFIG.SYS file on it. These lines must be in the. CONFIG.SYS:

```
USE OSRAM
DEVICE SPARTA OSRAM
DEVICE SIO
DEVICE ATARIDOS.SYS
```

The rest of this disk for anything you choose. To run AtariWriter Plus, boot the computer with this boot floppy disk in D1:. Remove the disk and insert the AtariWriter Plus disk and then type 'D1:**X AP.OBJ**'.

You may use a ramdisk at D3: - D9: with AtariWriter Plus, but you won't be able to get a directory of the ramdisk from the program. You can still use this for temporary storage.

**Note:** The above hints for AtariWriter Plus have been kept for historical reasons in this guide and still apply. For up to date word processing please check 'The Last Word' available from <http://atari8.co.uk/> as it especially supports SDX features.

## F System Drivers Summary

### Alphabetical quick reference

#### **ARCLOCK.SYS [/F]**

The driver for the battery backed 'Atari Real Clock'.

#### **ATARIDOS.SYS**

Reads and writes AtariDOS 2 disks.

#### **COMEXE.SYS**

\*.EXE files will be automatically executed 'through X'.

#### **CON64.SYS**

64-column text mode emulator.

#### **DOSKEY.SYS [d:][path][fname.ext]**

Extends the Command Processor with a history buffer, command aliases and a possibility to type multiple commands per one command line.

#### **ENV.SYS**

Keeps the environment variables in the extended memory.

#### **INDUS.SYS [n]**

Fast serial protocol engager for Indus GT, CA-2001, and LDW Super 2000 floppy disk drives. Also needed for Happy drives.

#### **IDEPTIME.SYS [/F]**

Driver for the realtime clock residing on the KMK/JZ IDE V 2.0 Plus interface.

#### **JIFFY.SYS**

Software clock driver (uses the VBI jiffy counter).

#### **PBI.SYS**

Buffers reads and writes to and from the PBI area (\$D800-\$DFFF).

#### **QUICKED.SYS**

Software screen accelerator for the 40-column text mode.

#### **RAMDISK.SYS [d:[,nbanks]] [/S]**

A ramdisk driver. '/S' forces loading the 'standard' 6502 RAM driver module even on 65816 machines.

#### **RTIME8.SYS**

The driver for the ICD R-Time8 battery backed clock.

#### **RUNEXT.SYS**

Extension to the Command Processor that allows to define associations between data types and application programs.

**SIO.SYS [/CA]**

The SIO driver.

**SPARTA.SYS [OSRAM] [nbufs[,nfiles]]**

The SD kernel and the main filesystem driver.

**ULTIME.SYS [/F]**

Driver for the realtime clock residing on Ultimate 1 MiB, SIDE 1&2, SDX SuperCart.

**XEP80.SYS [1|2] [/PN]**

The XEP80 driver.

**Z.SYS [/I|S]**

Installs a SD 3.2-compatible 'Z:' device.

Please check the SDXTK disk for more drivers.

## G Changes, Innovations And Way Ahead

Please find here an overview of the latest changes, innovations created, unsolved issues and ideas for future developments. The complexity of nowadays hardware and all the features asked for by the users of SDX lead to a continuing process of enhancement and development. The changes here relate to version 4.47.

If you should come across errors, problems, inconveniences or have an idea in mind, please do not hesitate to contact us at <http://sdx.atari8.info>.

### G.1 Kernel

- Corrected implementation of XIO 13 (FILE STATUS). Atari Artist now works.
- DLI in the custom NMI handler will now be handled only by returning from the interrupt immediately. That seems to be the safest way between ignoring completely and fully servicing (that last is risky, because the DLIV vector may be pointing to ROM, which is off during execution of the routine in question).
- Fixed the problem which caused the MACH.COM program to display bogus values for CPU speed percentage when in USE OSRAM mode.
- Putting the ECHO keyword before the USE keyword in CONFIG.SYS caused the USE to get ignored. Now USE is not strictly required to be the first keyword in CONFIG.SYS, it only must be executed before any DEVICE or SET.
- Added a new special build for MyIDE II with SDX stored in ROM (512 KiB), not in RAM (128 KiB) as in the also renewed regular build.
- Fixed an old ICD bug in the CIO interface, Atari Font Maker now works.
- Added handling of the DLI to the custom NMI handler (executed when the ROM OS is off). But it will now be handled only by returning from the interrupt immediately. That seems to be the safest way between ignoring completely and fully servicing (that last is risky, because the DLIV vector may be pointing to ROM, which is off during execution of the routine in question).
- Added the Super Cart build (for now: same as SIDE2, but pass-through).
- Small bugfix inside hwctrl procedure (may affect MEMLO on non-passthrough hardware).
- Improved compatibility of 1 MiB RAMBO and 576 KiB Compy Shop RAM extensions with the internal BASIC on XL/XE.

### G.2 Drivers and resident programs

- CHKDSK: the /V switch (and its code) has been removed, use CLX instead.
- In case of accessing file systems larger than 32 MB (like these handled by FATFS) now the IDE+2.0/U1MB PBI XDCB protocol will be used instead of the old IDEa extended SIO addressing protocol. In either case this only applies to the PBI devices, serial drives are unaffected.

- ENV will now occupy 14 bytes less of the main memory at some expense in the Ext RAM.
- SPARTA.SYS will now occupy 9 bytes less of the main memory (at expense of the Ext RAM).
- SIO2.OVL, SIO4.OVL and SIO5.OVL will now each occupy 3 bytes less in the main RAM (as above, at the expense of the Ext RAM).
- ATARIDOS.SYS: 6 bytes main saved using the same principles as above.
- TD:
  - 1) fixed a bug in displaying the time in the US format (am/pm);
  - 2) the seconds should now go more stable on NTSC computers;
  - 3) fixes made to prevent occasional stability problems, especially in USE OSRAM mode.
  - 4) I\_TDON will now return (in the accumulator) the previous state of the TD Line (0 OFF, 1 ON).
- SIOSET now accepts a new option WAITACK to define a custom SIO command response timeout (2 is the default). You can adjust it for devices that do not conform fully to SIO standards. For example "SIOSET WAITACK \$10" will handle SIO2BT device (without using customized OS routines).
- Fixed an inveterate bug in IDEPTIME.SYS (IDE+ 2.0 clock driver) which made reading NVRAM impossible.

### G.3 Library

- U\_GETADR can now parse expressions such as 'address+constant', 'address-constant', where the 'address' part is a numeric address value or a symbol, and the 'constant' part is a numeric constant.

Therefore the utilities which use this call (BLOAD, DPOKE, PEEK, POKE, MDUMP) will now allow syntax like: 'PEEK COMTAB2-4' or 'POKE 1536+32,1'.

The constant is only added to the lower 16 bits of the address, thus when the result exceeds 65535, the low word will wrap back to 0.

- Fixed a bug which caused the Sparta Commander to overflow the error trap stack after starting a handful of programs. The symptoms were various stability problems while using the Sparta Commander, especially hangs or reboots after a RESET.
- PRINTF now accepts strings longer than 255 characters. Also ASCII 0 may now be printed out by using the new escape sequence "\z".
- PRINTF has a new escape sequence \i which toggles the inverse video mode on an off for the strings being printed.
- Fixed a bug that caused GETDFREE to return improper values of X and P registers, when its execution succeeded.

### G.4 Formatter

- nothing new



## G.5 Utilities

- When the screen is 80-column wide, DUMP and MDUMP will now display the memory contents in rows containing information about 16 bytes each.
- MAN.COM will now silently ignore non-accessible paths included in the \$MANPATH (instead of exiting with an error).
- X.COM now sets the system variable RAMSIZ (\$02E4) along with the RAMTOP when disabling and enabling the SDX cartridge ROM. As a consequence, InterLISP/65 now starts without complaining that there is a cartridge inserted.
- Also a slight correction which prevents screen garbage which used to occur after this: POKE 106,PEEK(106)-4:GR.0:DOS made in TBXL.
- X.COM:
  - 1) fixed a bug that seriously contributed to the garbage visible on screen during loading programs while TD Line was active,
  - 2) old ICD bug fixed, CLSN Pascal now works without using an external loader,
  - 3) now sets the system variable RAMSIZ (\$02E4) along with the RAMTOP when disabling and enabling the SDX cartridge ROM. As a consequence, InterLISP/65 now starts without complaining that there is a cartridge inserted, even if there is none,
  - 4) also a slight correction which prevents screen garbage which used to occur after this: POKE 106,PEEK(106)-4:GR.0:DOS made in TBXL.
- CAR.COM:
  - 1) added support for the Weronika cartridge to run the dedicated programs, use CAR /F filename.ext,
  - 2) fixed a problem with calculating the hash value of the internal BASIC ROM on machines equipped with 576 KiB Compy Shop and 1 MiB RAMBO RAM extensions.
- RS232.COM program repaired, it did not work. Apologies.
- COMP will now
  - 1) display offsets counting from 0, not from 1,
  - 2) accept 24-bit offsets to both compared files, not 16-bit,
  - 3) use larger buffers and therefore is much faster,
  - 4) signalize EOF on either file instead of silently quitting.
- MDUMP has a new switch: /R when given will cause addresses to be displayed as offsets relative to the given starting address.
- LESS will now complain when the specified file does not fit in the buffer.

APPEND.COM will now consider the path separators '\' and '>' identical while doing string comparisons.

- CHKDSK.COM: the /V switch (and its code) has been removed, use CLX instead.

MDUMP.COM has a new switch: /R when given will cause addresses to be displayed as offsets relative to the given starting address.

- COPY -A will now not pick directories to be copied in "flat" mode (i.e. without recursing into subdirs). Previously specifying any attribute enabled the command to pick directories and copy them as files.
- COMMAND.COM:
  - 1) SET PROMPT=D\$N will now automatically display a colon after the drive number, as it is done after the drive letter for \$L,
  - 2) allow the user to switch to another default device than a disk: you can now set e. g. CAR: as the default device,
  - 3) when the default device is not a disk, its identifier will be automatically displayed before the unit number (\$N) or unit letter (\$L).
  - 4) corrected spelling of the "FTe" in the welcome message (thanks, Kyle!) in COMMAND.COM.
  - 5) ECHO FOOBAR >>FOO.BAR will now strip the extra blank it used to output at the end.
  - 6) ECHO /N FOOBAR >>FOO.BAR will also suppress the EOL.
- The BAT command GOSUB was broken, apologies. Fixed now.

## G.6 SDX Toolkit

- TAR 1.7:
  - 1) fixed reporting bogus error codes (like -245),
  - 2) fixed a problem with opening subdirectories for recursive scanning under SD 3.x,
  - 3) command line options may be now preceded with the slash character /, as in other SpartaDOS utilities; the - sign still works;
  - 4) the problems with I/O redirection (TAR /TV FOO.TAR >>FOO.TXT) should be now gone,
  - 5) corrections in date stamp conversion between the SD format and the Unix format,
  - 6) when no files match the given mask, TAR will now display a suitable message and do nothing instead of creating an empty archive containing nothing but the EOF mark (even though the latter thing is what the BSD TAR does in this situation).
- Sparta Commander now V. 1.0.0:
  - 1) added stripping trailing blanks in the command line before it gets passed on to the system,
  - 2) command lines will now be remembered by DOSKEY (if loaded).
  - 3) filenames without an ext will now get a dot appended,
  - 4) new user functions: Fmask, cHmod, mKdir, Rename,
  - 5) fixed a bug in Del,
  - 6) added a requester to Edit,
  - 7) better cooperation with RAM-disks,
  - 8) added some colors (on VBXE),
  - 9) added small, 24-hour clock display,
  - 10) SC will now respect the \$SCRDEF variable. Also a bugfix added in program's initialization. SAV file format changed, it should now be averagely several times smaller.
  - 11) the Help and Shift/Help keys are now doubled as as Ctrl/I and Shift/Ctrl/I (for 400/800, where there is no Help key on the keyboard);
  - 12) SC.INI has new keywords: EDIT= and VIEW=, with them you can define programs to be launched when the user selects Ctrl/E or Ctrl/V,

- 13) the macro processor can now substitute actual values for variables named in the macro string: %cs% (cursor selection), %csfile% (cursor selection, file), %csdir% (cursor selection, dir), %cdev%, %odev% (devices opened in active and inactive panel, respectively), %cpath%, %opath% (paths to directories opened in respective panels), %cmask%, %omask% (the respective file masks in both panels).
- 14) macros may now predefine an answer to the question "Run in 40-column mode?", if it is about to appear for the defined macro string,
- 15) the maximum number of macros increased to 13.

- CHKFAT.COM updated for XDCB usage (see above).
- New program: DU.COM, scans a directory (yes, recursively) and outputs information on the total size of the files and subdirectories inside.
- Fixed a minor bug in MKATR.COM, the program asked if to replace an already existing ATR file, but behaved the same regardless of the user's response.
- HRAMTST will now also test the linearity of a 64 KiB segment.
- HRAMSPD adds a check if the OS allows to switch the 65C816 into the native mode.
- FSTRUCT will now accept attribute parameters (+/-AHPS) before the file name so that you can use the program on a file which has e. g. +H set.
- XVER utility updated for MyIDE-II builds.
- XVER made work with old SDX versions 4.2x.
- Sparta Commander will now respect the \$SCRDEF variable. Also a bugfix added in initialization. SAV file format changed, it should now be averagely several times smaller.
- KTRACE: new utility to trace SDX system calls.
- RC\_GR8.SYS now version 0.86: MEMTOP 245 bytes higher (= 245 bytes more free RAM when the 80-column display is active), added font loading, scrolling the display horizontally (XIO 117 and 118), various fixes.
- INIDOS.SYS should now work for SIDE2.
- PCLINK: the /S switch removed, this mode was buggy and it anyways was an old test code which would not work correctly in some configurations (like USE OSRAM).
- CON.SYS: 80-column mode can now survive a warm reset.
- CON.SYS: fixed a bug which sometimes caused crashes when switching from a pixel mode to the 80-column console mode.
- EDDY.EXE: fixed a bug in the Search function, it now works as it should. Also fixed a bug causing the file selector to crash when accessing an AtariDOS/MyDOS disk. Few other minor problems cleaned up.

- Fixed a major bug in the S\_VBXE driver, which caused occasional hang ups after the Reset key was pressed.
- CHKFAT.COM: new command line tool to verify FAT file systems compatibility with our FATFS driver (see the archive with the driver).
- RAPIDUS.SYS and 65816.SYS drivers now version 1.2 (bugfix & size optimization).
- Sparta Commander now version 1.00
  - 1) added stripping trailing blanks in the command line before it gets passed on to the system,
  - 2) command lines will now be remembered by DOSKEY (if loaded),
  - 3) filenames without an ext will now get a dot appended,
  - 4) new user functions: Fmask, cHmod, mKdir, Rename,
  - 5) fixed a bug in Del,
  - 6) added a requester to Edit,
  - 7) better cooperation with RAM-disks.
  - 8) several bugfixes,
  - 9) added some colors (on VBXE),
  - 10) added small, 24-hour clock display,
  - 11) some other enhancements.
- RUN.COM: added extended memory support and 65C816 support.
- STAT.COM will now display the estimated size of the specified file or directory in logical sectors.
- New program: MD5 hash tool.
- MEMINFO.COM, a new tool that displays internal DOS' information about the extended RAM.

## G.7 To-Do List

- ARC has sometimes a problem packing very long files, i. e. the archive gets screwed. The problems seem to be limited to the Squeeze algorithm.
- CON64.SYS conflicts with CON.SYS.
- SIO.SYS does not handle properly transmission with speed index below 5. This depends on the hardware used. Most problems occur with SIO2SD and SDRIVE, while APE USB interface works with index of 2.
- LESS should be able to handle files being larger than free main memory.
- ACTION! monitor does not display commands if QUICKED.SYS is loaded.
- JIFFY.SYS, when running overnight, allows the date to turn e. g. to 31 November.
- Reading raw directory from CAR: reveals bogus file sizes.
- COMEXE will not work if wildcards are given in the extension.

- New text editor.
- Ramdisk for Turbo Freezer 2005 internal RAM.
- CALC.COM - command-line calculator which can assign the result to a env var.



## H Glossary

Terms and definitions important to A8 users as they apply throughout this guide and/or in many other A8 publications.

### A8

This is not a superior engine to a V8 but the short form for Atari 8-bit computer.

### Address

A location in memory. An address may refer to a RAM or ROM storage location, a hardware register for an external device or processor, or a combination of these through bank selecting. The A8 address range is from 0 to 65535 (\$0000-\$FFFF).

### Altirra and Atari800 emulators

Information to be added.

### Append

Simply to add to. To append one file to another is to add the first onto the end of the second. This is often used when loading device handlers.

### ASCII

The American Standard Code for Information Interchange. This code uses seven bit data words (0 to 127 decimal, \$00 to \$7F hex) to define a standard character and control set. For example, the character S is represented by the number 83 (\$53 hex).

### Assembly Language

A readable representation of code in machine language in which the CPU is programmed. 6502 assembly language uses three letter mnemonics and operands to represent the numbers of the instructions and arguments. This assembly language source code is then translated by an assembler to the numbers that the 6502 can understand. Advanced assemblers allow the source code to include labels, macros (collections of often used instructions) and more.

### ATASCII

The A8 version of ASCII. ATASCII uses eight bits, providing codes from 0 to 255 (\$00 to \$FF hex) and displayable characters for almost every code. There are several differences between ATASCII and ASCII, the most notable being that ATASCII uses 155 (\$9B hex) as an EOL (end of line) marker, while ASCII uses a 13 (\$0D) for CR (carriage return) and 10 (\$0A) as LF (line feed). Complete listings of ASCII and ATASCII values can be found in most programming reference books.

### ATR

The well known ATR 'Atari image' was created by Nick Kennedy, who invented the SIO2PC. Those files stored on other computer platforms use the extension ATR.

### AtraX SDX 128 cartridge

Information available via webpage of the "SpartaDOS X Upgrade Project".

### Axlon

Manufacturer of memory extension boards for the 400/800 computers. SDX handles Axlon compatible memory extensions up to 4 MiB (4,096 KiB).

**Bank**

Block of memory of specified size occupying a specific range of addresses. Bank select is the use of hardware registers to cause different banks of RAM, ROM, or hardware registers to occupy the same logical address space.

**Bank Switching**

The 6502 CPU can address 65536 memory locations. Additional memory beyond 64 KiB has to be bank switched. This technique allows to alter between main and extended memory. Currently SDX supports Axlon types for 400/800 up to 4 MiB, Port B types for XL/XE up to 1 MiB. Additional support is available for some 16-bit CPUs.

**Batch**

Is a text file containing a list of commands to be executed consecutively.

**Baud**

A unit of measure of serial data transmission, named after the French inventor Jean-Maurice-Emile Baudot. This is the number of code elements per second. While this term is used interchangeably with bits per second (bps) in common usage, it is not actually the same.

**Binary**

The base 2 numbering system. Each digit of a binary number can be only a 0 or a 1. This numbering system is used by all digital computers, since each digit may be represented by either the presence or the absence of voltage. Since binary numbers such as 01010011 can be tedious to work with, numbers are usually represented in decimal (83) or hex (\$53) format. A binary file is one that consists of numbers representing instructions and data directly readable by the computer. As interpreted by the Atari OS, a binary file also contains load addresses.

**Bit**

A single binary digit.

**Black Box**

PBI interface made by CSS with options and add-ons. Utilizes SCSI hard drives.

**Blitter**

This is a circuit dedicated to rapid movement and modification of data in the memory. It copies a huge amount of data from one memory area to another quickly. It works parallel with the CPU freeing it up from basic workload operations.

**Boot**

The initialization of the computer, caused either by turning it on or by executing a cold start. The only difference in the two is that it is usually possible with SDX to preserve ramdisk contents after a cold start but not after losing system power.

**Buffer**

Memory area used as a temporary storage area for data. Buffers are commonly used for I/O involving the keyboard, screen, disk drives, etc.



**Byte**

A binary number consisting of 8 bits. Since the 6502 processes 8 bits of data at a time, most data in the A8 is represented as one or more bytes. Each address points to one byte. These values are most easily referenced as a two digit hex number.

**Centronics**

A standard parallel interface named for the company that first used it. Almost all parallel printers use 'Centronics' type ports with a 'standard' 36 pin connector. The A8 can be connected to a printer with a 'Centronics' port only via an appropriate cable and interface providing a parallel port, such as the Atari 850, the P: R: Connection, the Printer Connection, the Multi I/O or similar.

**CIO**

Central Input/Output. All communication with the screen, keyboard, and all peripheral devices may be handled through this part of the Atari OS. The CIO is one of the things that sets the A8 OS above other 8-bit computers.

**Cold Start**

To cause the computer to initialize as if power were removed and reapplied without actually turning the computer off. This is faster than cycling power and will in most cases allow ramdisk contents to be preserved with SDX.

**Command**

An instruction given to the computer from the user.

**CP**

Command processor. This portion of the DOS environment provides the interface between the user and the DOS. The CP prompts the user, interprets the commands given and causes the specified operation to be executed.

**CPU**

Central Processing Unit. This is the main part of the computer, the part that reads and executes instructions. All programs must be translated into commands and data that the CPU can understand. The standard CPU for the A8 is the 6502. However, there are customized A8 machines using 16-bit CPUs, e. g. 65C816).

**CRC**

Cyclic Redundancy Check. This is a two byte number produced by performing a complex mathematical operation on a set of data. CRC is used in many applications, e. g. file transfer protocols and the ARC program.

**Current Directory**

The directory assumed if none is specified. The default is the main, or root directory. The current directory may be changed to any subdirectory on a disk with the CHDIR command.

**Cursor**

The mark on the screen that points to the place where the next action will take place.

**Cylinder**

Used interchangeably with track, most often with hard drives, since these have multiple surfaces and heads.

**Data**

Information used or processed by a program.

**Debug**

To isolate and correct errors in a program.

**Decimal**

The base 10 numbering system. This numbering system is used by human beings everywhere, consisting of numbers made of digits ranging from 0 to 9. While easier to understand, decimal numbers can be awkward to use for computer purposes.

**Default**

The value or condition assumed if none is specified.

**Density**

Generally, this is the number of bytes in each sector of a A8 disk. Single Density refers to 128 bytes per sector, while Double Density refers to 256 bytes per sector. DD 512, the new density introduced by SDX 4.4x, refers to 512 bytes per sector.

**Device**

An input and/or output interface to the computer, whether an actual physically external device, such as a printer, or a part of the computer simulating an external device, such as the screen editor. A program may access the D; E; S; R; P; C; K; (disk, screen editor, screen, serial port, printer, cassette drive, and keyboard, respectively) and other added devices through the CIO. SDX, through the 'kernel', uses the devices DSK, CLK, CAR, CON, COM, PRN, NUL, PCL. These kernel devices may be accessed through the D: device from the CIO or independently through the command processor.

**Directory**

The list of files and subdirectories on a disk or in a virtual disk (such as a ramdisk or the CAR: device). If there are subdirectories on the disk, then it is the list in a given directory.

**DOS**

Disk Operating System. The program that manages disk I/O to and from the computer. In practice, most DOS types also add functions and features to the operating system of the computer in areas not directly related to disk I/O.

**DMA**

Direct Memory Access. Any part of the computer system that can directly address system memory is considered to be a DMA device. The CPU is, of course, a DMA device. The only other DMA device in the A8 system is ANTIC, the graphics coprocessor.

**DRAM**

In most computers the memory is Dynamic Random Access Memory. It leaks charge, which causes the loss of stored information. Therefore the charge needs to be refreshed periodically. Thus it is called a dynamic memory as opposed to static and flash memory. DRAM loses the stored information at power removal.

**Driver**

A program that usually remains in the computer and handles a specific device or operation. For example, SDX includes a driver to allow it to read from and write to disks in AtariDOS 2 format .

**ECI**

Enhanced cartridge interface to be found with the A8 XE series. It adds the needed signals to the cart port to provide a bus interface like the PBI on XL computers, but provides more options.

**Emulator**

Software, Hardware or a combination of both. Allows a computer to act like another machine and enables the use of software originally written for another system.

**File**

A collection of information, usually stored as a named unit on a mass storage medium for computers.

**Filespec**

Describes the specification of a file and its place within the SDFS. Device identifier and pathname make up the filespec.

**Firmware**

Software that is permanently encoded into ROM, preventing it from being changed or erased. SDX is firmware, as are the computer's OS, the US Doubler and any cartridge-based program.

**Flash Memory**

This is a non-volatile computer storage chip that can be electrically erased and reprogrammed. No power is needed to maintain the information stored in the chip. Used in newer A8 devices.

**FM - Frequency Modulation**

Is a method of encoding digital data on magnetic media, that was used with early hardware like A8-compatible floppy disk drives.

**GZIP**

Short form of 'GNU zip'. It is a compression utility adopted by the GNU project.

**Format**

To initialize a medium so that it may be used to store and retrieve information. Physically the magnetic medium on the disk or hard drive is structured into tracks, which are divided into sectors. Directory information is usually written to a disk after it is physically initialized as part of the format operation. The SDX command to format a medium is called FORMAT.

**Handler**

An often memory resident program that handles a device. This usually patches into the CIO handler table to allow applications programs to access the device as it would any other. For example, most DOS types add a D: device handler. An R: device handler must be added for most serial I/O applications.

## Hard Copy

Information printed on paper.

## Hard Disk

A high capacity disk drive. Hard disks are usually sealed units, storing anywhere from 5 megabytes to several terabytes of information. Storing data on and retrieving data from a hard drive is usually many times faster than performing the same functions on a floppy disk drive, especially if the hard disk is connected to the computer through the PBI or ECI. On hard disk interfaces adapters for Compact Flash (CF) or Secure Digital (SD) cards can be used as mass storage.

## Hardware

The computer, peripherals and all related circuitry. Generally, any electronics that you can touch can be considered to be hardware.

## Header

Data at the beginning of a file that provides information about the type of file, how it should be run and where it should be loaded.

## Hex

Short for hexadecimal, the base 16 numbering system. It consists of numbers made of digits ranging from 0-9 and from A-F (representing 10-15). This is the easiest and clearest way to represent the eight and 16 bit binary numbers used with the A8. To differentiate them from decimal numbers, hex numbers are usually preceded by a \$ character.

## High Density Interface (HDI)

Interface for connecting up to 4 floppy disk drives to the A8 - mix of 3.5" and 5.25" drives is legal. The drives need to support the disk change signal at pin 34 of the Shugart bus. Maximum transfer rate: 500 Kbit/s. Not supported: Atari Medium Density and 1.2 MB floppy disk drives.

## I/O

Input/Output. This refers to communication between the computer and the real world, including all devices and peripherals.

## ICD

The company that had written sophisticated software and designed power peripherals for the A8 from 1984 to 1993. SD, SDTK, SDX, US Doubler, P: R: Connection, Printer Connection, Multi I/O, R-Time 8, and more. ICD does not serve A8 anymore.

## IDE

Integrated Drive Electronics is a standard comprising a connector, an interface definition and a drive controller integrated into the drive itself. The interface connects to a parallel ATA drive. The integrated controller presents the drive as an array of 512-byte blocks to the host computer utilizing a simple command interface. All operation details of the drive are handled by the drive's controller. The host computer just has to ask for a specific sector or block to be read or written to. Several IDE interfaces have been developed for A8. They attach either to the parallel bus interface, the cartridge interface or the SIO interface connector. On IDE interfaces adapters for Compact Flash (CF) or Secure Digital (SD) cards can be used as mass storage.

### Image

This is a software copy of a physical disk/partition. It contains the entire data from the source media, including the file structure and all files and folders in one single file. Image files store data in a raw, binary format. So they do not have a file system to tell the computer how to access the files and folders in the image. To access the data in an image it must first be mounted by special software.

### Incognito board

Information available via webpage of the "SpartaDOS X Upgrade Project".

### intSDX128 and intSDX128 "flash"

Information available via webpage of the "SpartaDOS X Upgrade Project".

### IOCB

Input/Output Control Block. A sixteen byte block of memory used to pass parameters to and from the CIO for I/O functions. There are 8 IOCBs starting at \$0340, numbered 0 through 7. IOCB 0 is normally used for the screen editor. A different IOCB must be used for each open device or file.

### K, KB or KiB

K or KB were used to name the memory size in kilobytes at the times when the A8 was invented, which actually is misleading since kilo means 1,000 and not 1,024. According to the IEC definition from 1998 in binary use  $2^{10}$  bytes got the prefix 'Kibi' (abbr. KiB). For A8 systems 1 K, 1 KB or 1 KiB is 1024 bytes or  $2^{10}$  bytes.

### Kernel

The center of SDX, responsible for I/O functions. The kernel provides I/O independent of the CIO and many new devices, such as CAR:, CON:, PRN: and DSK:.

### Kludge

A program or part of a program or a hardware assembly that produces the desired results but does so in a complicated and/or unnecessary way.

### KMK/JŽ IDE / IDEa / IDE Plus 2.0

Information available via webpage of the "SpartaDOS X Upgrade Project".

### Language

A program or development system that provides an easier or faster way to write a program. A program written in a programming language is then either translated into machine code (compiled) or is used as a set of commands for another program (interpreted) which then performs the desired tasks. ACTION!, BASIC, Pascal and C are examples of popular A8 languages.

### M, MB or MiB

See K, KB or KiB for details.  $2^{20}$  bytes or  $1,024^2$  KiB.

### Machine Code

The program that the CPU reads and understands. All programs written in other languages must be translated into 6502 machine code (often called machine language) before they can be run. While the terms assembly language and machine language are often used interchangeably, they are not the same.

Maxflash 1 Mib, 8 MiB  
NIL.

Maxflash MyIDE+Flash  
NIL.

#### Medium/Media

Non-volatile storage devices on which A8 file systems can be created and used. These are disks, hard drives, CD ROM, memory cards or sticks and with emulators just 'software devices' on other systems.

#### MEMLO

The address of the start of usable memory above DOS, memory resident handlers and other programs. The number is stored at memory locations 743 and 744 (\$2E7 and \$2E8). When using BANKED memory, SDX provides the lowest MEMLO possible, meaning that you have more programming space available.

#### Memory Resident

A program that remains in memory after being run and continues to perform its task or tasks when necessary. DOS is memory resident, as are added device handlers. Most memory resident programs relocate to low memory and protect themselves from being overwritten by raising the low memory pointer.

#### MIO (MULTI I/O)

An interface invented by ICD that connects to the PBI and provides up to 1 MiB of RAM, an RS232 serial port, a parallel printer port and a SCSI/SASI compatible hard disk port. The MIO allows reassigning the logical drive units of physical floppy disk drives, hard drives and ramdisks, so that the computer can be booted from any of these. It has been re-issued and is available from <http://www.rasterline.com/>.

#### MFM - Modified frequency modulation

Is a method of encoding digital data on magnetic media, that was used with early hardware like A8-compatible floppy disk drives, other disk drives and small hard disk drives up to 5 MB. As an enhanced frequency modulation (FM) encoding scheme it doubles linear bit density and decreases flux reversal density without increasing recorded magnetic density.

#### Modem

MOdulator/DEModulator. A peripheral device that translates serial data from a computer into sounds that may be transmitted over telephone lines, allowing communication with similarly equipped computer systems at a distant location. Some MODEMS, such as the Atari 1030, XM301, and SX212 may be connected directly to an A8, but most require a serial interface such as the Atari 850 or the ICD P:R: Connection. Hayes compatible MODEMS accept a standard set of simple commands to perform a variety of tasks, such as dialing, answering the phone, hanging up, etc.

#### MSC

IDE interface as a parallel bus device for ATARI XL. Connects IDE hard drives or CD-ROM drives to the XL. Uses advanced drive mapping for 'on the fly' drive exchange.

MULTI I/O → MIO

**Multiplexer**

Very advanced network system from CSS, nowadays available from ILS. A master hosts everything for up to eight slaves. Connections are made via cart port providing full system speed to exchange data among the A8s in the network.

**MyIDE II**

Information available via webpage of the "SpartaDOS X Upgrade Project".

**Nibble**

Four bits, or one half of a byte. A nibble can be represented by a single digit hexadecimal number.

**NVRAM**

Non-volatile random access memory is RAM, which preserves its memory content when power is turned off. Please see DRAM and SRAM for the differences. The most common form of NVRAM is flash memory. Some new A8 devices use it.

**Optimize**

Some ancient hard drives are physically formatted so that the very first sector of a partition has number 0, and not 1, as it should be on an Atari hard disk. This sector 0 is not accessible, but does count into the summary of existing sectors returned to the computer by the drive controller. On such a drive, the sector 'reclaimed' by the Optimize function in the formatter menu does not really exist. Keep Optimize off when building directories on such a drive. E. g. hard drives from Supra Corp. and K-Products work like that.

**OS**

Operating System - a software that handles hardware and software resources in a computer system. It also provides common services like disk I/O for applications. The OS is the core component of the system software for a computer system.

**Parallel**

The transfer, processing, or manipulation of all the bits in a byte simultaneously by using a separate line for each. This is usually faster than serial. Most printers are parallel devices (→ Centronics).

**Path**

A list of subdirectory names describing the course from either the root directory or current subdirectory to a specific subdirectory.

**PBI**

Parallel Bus Interface, the large connector on the rear of the 600XL and 800XL, allows the A8 to communicate quickly with powerful external devices. The PBI can be duplicated on the 130XE by a simple adapter connected to the cartridge and the ECI ports. Fast mass storage is usually connected to the PBI. A maximum of eight devices can be attached.

**PERCOM**

The PERCOM block used with storage devices (except cassette recorders) consists of 12 bytes of data. It describes the physical format of a formatted disk/partition and tells a drive about the physical format that should be applied to the medium when performing the succeeding format command.

### Peripheral Device

Literally, a device on the periphery. In the computer world, a peripheral device, or just peripheral, is hardware added to the basic system configuration. Floppy Disk Drives, Printers and MODEMs for example, are peripheral devices.

### PIPE

A pipe allows to redirect output of a program as input for another program. Unique feature of SDX on the A8 platform.

### Port

A place of access to a system; e. g., the joystick port, the parallel bus interface port, the serial I/O port, cartridge port, etc.

### Program

A set of instructions that cause a task to be performed by a computer. Programs must adhere to the order and conventions of the language in which they are written.

### Prompt

A signal to the user that some action may be required. The D1: or A: prompt with SDX tells you that the computer is ready for input.

### RAM

Random Access Memory. The storage area to which the computer save and retrieves from information. RAM chips in A8s are DRAM.

### RAMDISK

A specified area of memory that simulates a disk drive. A handler or driver is used to make this memory appear to be a disk drive. Depending on the type of memory chips (DRAM, SRAM, FLASH MEMORY) they will lose or keep the contents when power is removed. Newer A8 devices often use SRAM or flash memory.

### Real Time

Relating to real world time. A real time clock uses the actual time. Real time can also refer to things that occur at the same time or at realistic speeds.

### Redirection

Allows redirection of inputs or outputs depending on the systems capabilities.

### Relocatable

A program that can be moved to different areas in memory and still operate properly. SDX is relocatable. Most SD drivers and handlers are self-relocating, usually meaning that they relocate themselves to low memory and move the low memory pointer MEMLO just above the code.

### ROM

Read Only Memory. ROM is like RAM except that it can not be changed and will remain the same even after a loss of power. With the original SDX from ICD, all of the programs in CAR; and the Atari OS are in ROM. Nowadays other memory chips are used as well to store those information, e. g. SRAM or Flash Memory.



**RS-232**

A standard serial communications interface documented by the Electronics Industries Association. Most MODEMS use an RS-232 interface. The A8 does not have an RS-232 port, but one may be added with an Atari 850 interface or a P: R: Connection.

**SCSI**

Small Computer System Interface. Many hard drives use a SCSI or SASI (Shugart Associates System Interface) bus. Up to eight SCSI devices may be connected to a SCSI port on a computer. The Multi I/O (MIO) from ICD and The Blackbox from CSS provide a SCSI port for XL/XE computers.

**SDX 128 "flash" cartridge**

Information available via webpage of the "SpartaDOS X Upgrade Project".

**SpartaDOS X cartridge from ICD - Upgrade**

Information available via webpage of the "SpartaDOS X Upgrade Project".

**Sector**

The standard block of storage used on disks. Sectors on A8 storage media may contain 128, 256 or 512 bytes.

**SIC! Cartridge**

Information available via webpage of the "SpartaDOS X Upgrade Project".

**SIDE, SIDE2**

Information available via webpage of the "SpartaDOS X Upgrade Project".

**Skew**

Helps to accelerate read/write processes on real floppy disk drives only. It refers to the order in which the sectors are arranged in a track. The optimum skew puts sectors in an order that after sector 1 is accessed and the drive CPU is ready for the next one, sector 2 is directly under the head for processing. Then sector 3, 4 and so on. Usually 2-8 sectors have passed under the head before the next sector can be accessed. This varies with drive speed and SIO baud rate. If the skew is off, it may take a full disk revolution before the next sector can be read or written to. The drive will be remarkably slow then.

**Serial**

The transfer of information on one signal line, one bit at a time. Most MODEMs and all devices connected to the serial I/O port (SIO) on the Atari are serial devices.

**SIO**

Serial Input/Output. All communication with devices connected to the serial bus of the A8 are handled through this routine. Devices on the parallel bus that simulate SIO devices are also accessed through the SIO. The Atari OS provides an SIO routine. SDX, however, uses its own SIO code.

**SIO2PC**

The most spread type of all emulated peripherals for A8. A hardware/software combination invented by Nick Kennedy utilizing a PC via the serial port as peripheral. On the PC a program maintains Atari images (→ ATR) as 'disk drives'. Similar developments working alike are SIO2PC(USB), SIO2BT (bluetooth device).

## Software

Programs, documentation and data files that allow a computer to function.

## Sparta

A city-state of ancient Greece; synonym for power!

## SRAM

Static RAM unlike DRAM does not need to be periodically refreshed. SRAM also loses data at power removal. Some memory extensions for A8 utilize SRAM.

## Subdirectory

An additional directory on a disk allowing better organization through the grouping of associated files. Subdirectories are treated as entries in existing directories.

## Syntax

The order and wording of commands or statements.

## TAR

Origins from early UNIX as tool to handle tape archives. Today TAR collects many files into one larger file for distribution or archiving. It preserves the directory structure with all files, but does no compression. Usually, on other systems than A8 GZIP is used to compress/decompress TAR archives.

## TPI

Tracks Per Inch. This indicates how densely data can be packed on a diskette. TPI tells how many tracks cross a one inch segment of the radius of the diskette.

## Track

A circular section of a disk surface. Each track is divided into sectors. A standard A8 floppy disk is formatted into 40 concentric tracks each containing 18 sectors.

## Truncated

Shortened. Usually used when the provided amount of data exceeds the expected amount and what can not be accepted is discarded.

## Turbo Freezer 2005

Information available via webpage of the "SpartaDOS X Upgrade Project".

## Ultimate 1 MiB

Information available via webpage of the "SpartaDOS X Upgrade Project".

## UNIX

Originally named UNICS - Uniplexed Information and Computing System - this OS had big influence on development of many other operating systems; also on SDX 4.4x.

## Variable

A symbol that represents a quantity that is changeable or has no fixed value.

## VideoBoard XE

Video hardware add-on for A8 types XL, XE and XEGS. Features RGB output with up to 1024 colors on screen from 21 bit palette, graphics resolution up to 640x480i (640x240p) in 64 colors, 320x240p in 1024 colors, and 160x240p in 1024 true 80 char

mode for text display, blitter (→ Blitter) with 7 modes of operations capable of zooming displayed data, transparency, collision detection and many other features, full downward compatible with GTIA chip and many, many more specialties.

#### Virtual Disk

Something that appears to the system and user to be a disk drive but is not. Ramdisks, both internal and external, ROM disks and the CAR: device in SDX are examples of virtual disks.

#### Warm Start

A system reset that does not clear out all memory as a cold start does. Warm starts do reset several system pointers.

#### Wildcards

A symbol that is used as a substitute for one or more characters in a file or directory name to allow more than one file to be selected. '\*' and '?' are the two valid wildcards with SD and most others.

#### Word

In the most common usage, a word is a sixteen bit or two byte number.

#### XIO

Extended Input/Output. A general I/O statement available in most Atari languages that allow CIO operations to be performed that are not supported by specific commands.



## Table of Figures

### Screenshots

Fig. 1: Boot Information - 48 or 64 KiB of memory .....	3
Fig. 2: Boot information - 128 KiB memory .....	4
Fig. 3: Boot information - sufficient memory found .....	4
Fig. 4: Formatter Menu .....	5
Fig. 5: First Formatted Disk .....	5
Fig. 6: Create new files on test disk .....	6
Fig. 7: new files on test disk .....	6
Fig. 8: Time and date display line .....	7
Fig. 9: Erase Files .....	8
Fig. 10: Copy Files .....	9
Fig. 11: Use of wildcards for Directory Listing .....	10
Fig. 12: Use of wildcards when copying .....	11
Fig. 13: Use of change directory .....	11
Fig. 14: Create New Batch File Using ED .....	13
Fig. 15: Save New Batch File in ED .....	13
Fig. 16: Append a temporary path .....	27
Fig. 17: Change time and date of a file .....	42
Fig. 18: Change volume name .....	43
Fig. 19: Compare given files .....	47
Fig. 20: Using the DATE command .....	54
Fig. 21: Check the installed system devices .....	57
Fig. 22: Display the properties of drives found .....	58
Fig. 23: Display the directory .....	59
Fig. 24: File dumped to the screen .....	61
Fig. 25: Small text editor .....	64
Fig. 26: Find files on the system drives .....	66
Fig. 27: Formatter menu .....	68
Fig. 28: KEY buffer is ON .....	72
Fig. 29: Less used to format a directory output .....	74
Fig. 30: Available Man Pages with SDX .....	76
Fig. 31: Memory dump .....	80
Fig. 32: Memory information .....	81
Fig. 33: Detailed memory information .....	82
Fig. 34: Additional memory information on 65C816 high ram .....	83
Fig. 35: Menu program - File Commands .....	84
Fig. 36: Menu program - Directory Commands .....	86
Fig. 37: Menu program - Xtra commands .....	87
Fig. 38: Customize your system prompt .....	95
Fig. 39: Display the current working directories .....	96
Fig. 40: RENDIR Command .....	98
Fig. 41: SAVE Command .....	101
Fig. 42: Display the system variables .....	102
Fig. 43: SWAPPING logical drives .....	107
Fig. 44: Using the TIME command .....	109
Fig. 45: TYPE used to display the file MAN.MAN. ....	110
Fig. 46: Restore files using UNERASE .....	111
Fig. 47: Display the SDX version information .....	112

Fig. 48: CLX 1.9b checking DD 512 medium .....	200
Fig. 49: Edit Screen Eddy 2.01 .....	201
Fig. 50: Sparta Commander 1.00 .....	202
Fig. 51: SDX Imager - User Bank Content .....	227

# Index

## In alphabetical order

- Batch Files,
  - AUTOEXEC.BAT, 116
  - Comparisons, 118
  - Conditionals, 116
  - EXIT, 121
  - GOSUB, 121
  - GOTO Jumps, 119
  - INKEY Command, 120
  - Loops, 118
  - Parameters, 116
  - Procedures, 120
  - SETERRNO, 121
  - Syntax, 115
- Command Processor,
  - CP interface, 3
  - I/O Redirection, 121
  - Pipes, 122
  - Prompt, 3
  - Search Path, 122
- Commands,
  - Alias, 25
  - Availability, 26
  - Command Length, 18
  - Device Identifiers, 18
  - External, 25
  - Filespec, 18
  - Internal, 25
  - Names, 25
  - Parameters, 25
  - Related, 25
  - Remarks, 26
  - SDX commands, 25
  - Syntax, 25
- Configuring The System,
  - Bypass CONFIG.SYS, 159
  - Character Sets, 161
  - Config Selector, 161
  - CONFIG.SYS, 157f.
  - DEVICE, 159
  - MERGE, 159
  - SDX Imager.86, 158
  - SET, 159
  - USE, 158
- Directories,
  - Entries, 17
  - Subdirectories, 17
- DOS,
  - AUTOEXEC.BAT, 103, 116
  - BASIC.SAV, 33
  - CAR.SAV, 37
  - Cold Start, 34
  - CONFIG.SYS, 82, 116
  - Density,
    - DD 512, 69
    - Double, 69
    - Dual, 69
    - Enhanced, 69
    - High, 69
    - Medium, 69
    - Single, 69
  - directory format, 6
  - DOS limitations, 231
  - Error Messages, 233
  - Introduction, 3
  - MEM.SAV, 33, 37
  - RAMDISK, 4
- Drivers, 157, 165, 187
- Emulators,
  - Altirra, 22
- Filenames,
  - Basic Form, 15
  - Extensions, 15
  - Wildcard Characters, 16
- Glossary, 265
- Hardware,
  - 65C816, 192, 199
  - ATARI Computer,
    - 1200XL, 248
    - 400, 3
    - 800, 3, 22
    - XL/XE, 3
- Disk Drives,
  - 1050, 224
  - 1050 Turbo, 71
  - 810, 224
  - ATR8000, 70
  - CA-2001, 71, 168
  - Happy, 71, 168
  - HDI, 70f.
  - High Speed, 69
  - HyperXF 551, 71
  - Indus GT, 69, 71, 167
  - LDW Super 2000, 71, 168

- PERCOM controller, 70
- Rana 1000, 71
- Speedy, 71
- TOMS 710, 70
- TOMS 720, 70
- TRAK-AT, 71
- Ultra Speed, 69
- US Doubler, 69, 71, 167
- XF551, 69, 71, 223
- Indus GT, 195
- Memory,
  - RAM, 3
  - Ultimate 1MB, 161
- PBI hard drive, 157
- PBI IDE Controller,
  - Blackbox, 79, 107
  - IDE Plus, 70
  - IDEa, 70
  - KMK/JŽ IDE 2.0 Plus, 70, 161, 167, 171
  - MIO, 79, 107, 167, 169
  - MSC, 79, 107, 167
- Rapidus Turbo Board, 191
- Realtime Clocks,
  - R-Time 8, 22
- VBXE, 71, 114, 188f., 202
- Path,
  - Length, 18
  - Pathnames, 17
- Programming,
  - ATR, 134
  - Bit Map, 154
  - BOOT, 134
  - CHDIR, 133, 137
  - CHKDSK, 135
  - CIO, 125
  - Decoding the drive identifier, 142
  - Default Drive, 125
  - Direct Disk Access, 156
  - Directory Formatting Attributes, 127
  - Directory Structure, 154
  - Erase File(s), 129
  - FORMAT, 135
  - Get File Length, 132
  - LOAD, 132
  - MKDIR, 133
  - NOTE, 131
  - Open File, 126
  - Page Seven 'Kernel' Values, 145
  - PERCOM, 155
  - POINT, 130
  - Protect File(s), 129
  - Raw Directory, 128
  - Rename File(s), 129
  - RMDIR, 133
  - Scrolling the display up, 149
  - Sector Map, 154
  - Sparse Files, 131
  - Symbols, 143
  - Unprotect File(s), 130
  - User Accessible Data Table, 139
  - Using the CON: Drivers, 147
  - Vectors Under the OS ROM, 143
  - XIO statements, 125
- Programming Languages,
  - ACTION!, 125, 233, 252, 262
  - BASIC, 6, 125
  - BASIC XE, 253
  - BASIC XE Extensions, 253
  - MAC/65, 101
  - Turbo-BASIC XL, 251
- SpartaDOS Toolkit, 66
- SpartaDOS X Toolkit, 26, 183
- Symbol, 25
- Word Processors,
  - AtariWriter Plus, 254
  - The Last Word, 254







# SPARTADOS X

## User Guide

This is the most advanced disk operating system ever designed for your ATARI 8-Bit computer. While still supporting the first generation machines ATARI 400 & 800 with at least 48 KiB of memory, SpartaDOS X supports all modern devices and gadgets available for the ATARI 8-Bit line of computers. And it is a living, constantly developing project.



This manual provides you with an in-depth look into the sophisticated world of SpartaDOS X, from the basics of DOS to a technical analysis of the disk and file structure utilizing hard drive partitions up to 32 MiB in size. The manual as well includes a wealth of information on the command set, batch files, I/O redirection, SIO high speed up to 125 KBit/s, search paths, programming, and more. You may take full advantage of the outstanding power of SpartaDOS X and its features in little time and perform really complex tasks with ease and configure your system for optimum use.



Not only does SpartaDOS X cater to the entire 8-bit line, but also employs full use of extended memory up to 16 MiB for different purposes including ramdisks. The high speed operation has been re-written to support newer drives when teamed up with. Quite a couple of new drivers, tools & utilities, a re-designed memory management, the SpartaDOS X Toolkit and SpartaDOS X Add-Ons let you get more out of the system while using the new SpartaDOS X. Last but not least support is also there for special hardware like the VBXE or the new Rapidus Turbo Board. An additional source, a Programming Guide, is available for those who want to write their own SDX applications or programs.

DLT  
Team

# SPARTADOS X User Guide

SDX  
4.48