SpartaDOS X

For ATARI 8-Bit Computers only

DLT

Latest Edit: 7 July 2013

We are proud to present the enhanced and remarkably upgraded

SpartaDOS X version 4.46

The most advanced Operating System
for ATARI 8-Bit Computers

SpartaDOS X V. 4.17 was the first release in late 1988 by ICD. Last known commercial version 4.22 by FTe dates Nov 5, 1995. A decade later this abandoned product got revived by its enthusiasts, who developed it to a new level. The new release again proves the power of SpartaDOS X, and its unique status in the ATARI 8-bit world.

SpartaDOS X is now available for almost every ATARI 8-bit platform including emulators and the brand new KMK/JZ IDE V2 plus. Please see annex G for updates and changes in relation to the previous versions.

And there is more to explore reading the manual and using SpartaDOS X.

A good reference is also *http://sdx.atari8.info/* for more information, technical details, add-ons, toolkit and enhanced support.

Take this manual for courtesy and keep ATARI 8 bit computers alive!

June 2013

CREDITS

based on works done by	:	Prof!, MMMG, DLT Ltd.
new code and design	:	DLT Ltd.
emulator version	:	dwhyte
hardware	:	Pasiu/SSG, Jad, Zenon/Dial, DLT Ltd., MetalGuy66, Candle, Simius
hosting	:	krap.pl
devtools	:	DLT Ltd., Tebe/Madteam, others
manual	:	Mikey, dely, DLT Ltd
other support	:	ABBUC, Epi/TRS, Krap, Mikey, Pin/TRS, Electron, Stephen Carden, sup8pdct, Tomo Hornacek, Guus Assmann

SpartaDOS X

The Most Powerful 8-Bit
Disk Operating System

Original by ICD
Enhanced Version by DLT Ltd.

Note - throughout this manual:

SpartaDOS, SpartaDOS Construction Set, SpartaDOS X, SpartaDOS Toolkit, FlashBack!, Multi I/O, MIO, P: R: Connection, Printer Connection, U1traSpeed, US Doubler, ACTION!, MAC/65, BASIC XL, BASIC XE, OS/A+, and DOS XL are trademarks of FTe - Abandonware.

ATARI 130XE, 65XE, 800XE, 800XL, 400/800, 810, 850, 1050, XF551, XE Game System, XEGS, XEP80, ATARI DOS 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 trademark of AT&T.

ARC is a trademark of System Enhancement Associates.

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/JZ IDE V2 Plus is Copyright © 2011 by Konrad Kokoszkiewicz & Jacek Żuk
----

SpartaDOS X - Published by ICD, Inc., 1988.
Last commercial version by FTe, 1995.
New Enhanced Version by DLT Ltd. 2008 - 2013

Table of Contents

1 Introduction

2 An Introduction to SpartaDOS

3 SpartaDOS X Overview

4 The Command Processor – Commands

5 The Command Processor – Advanced Features

6 Programming with SpartaDOS X

A DOS limitations

B Error Messages

C Command Summary – Alphabetical

D Command Summary – By Function

E Miscellaneous Notes

F System Drivers Summary

G Enhancements And New Features

Table of Figures

Index

1 Introduction

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

If this is your first experience with any version of SpartaDOS, it would be wise to follow the step-by-step examples presented in **Chapter 2.** This will help you to learn how SpartaDOS X operates and how you can take advantage of its power.

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

Chapter 4 describes the SpartaDOS X 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 SpartaDOS X, 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, including simple BASIC statements for novice programmers and detailed machine language access to the inner workings of SpartaDOS X for advanced programmers. Examples are given in BASIC and assembly language to aid the programmer in integrating these concepts into the own programs. A programming manual is already available in Polish language.

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

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

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.

SpartaDOS X is by far the most complex disk operating system available for the 8-bit ATARI. The very things that make SpartaDOS X 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 SpartaDOS X or any other former ICD 8-bit product, we advise you to post respective questions on supporting ATARI 8-bit fora on the Internet. You will find all information about the SpartaDOS X project on its homepage at

http://sdx.atari8.info/

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

Also patched or modified operating systems (OS) may hamper the use of SpartaDOS X on your machine. If you experience problems try an unmodified ATARI OS first.

Technical Advice: The full potential of SpartaDOS X is available only on XL/XE computers. However, it can also be used with ATARI 400/800 machines having at minimum 48 KB of RAM considering the differences.

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

Note: All information about SpartaDOS X provided in this manual refers to the SpartaDOS X cartridge. In general the manual applies to other hosting platforms or emulators as well. Please refer to the respective manual, if you should note differences or technical limitations.

2 An Introduction to SpartaDOS

This chapter is specifically for those of you who have no prior experience with SpartaDOS and may be somewhat confused by all of this. If you are a user of SpartaDOS 2.x or 3.x, you may wish to simply review this chapter and move on to chapter 3, which outlines the new features found in SpartaDOS X.

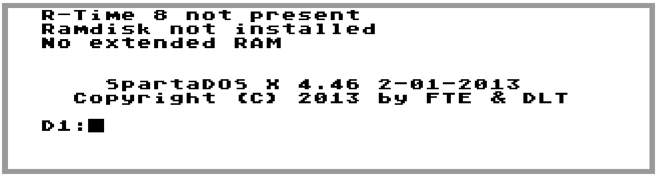
What is DOS?
DOS stands for Disk Operating System. The primary purpose of DOS is to allow the computer to communicate with one or more disk drives. In practice, a DOS provides many more useful features.

SpartaDOS
SpartaDOS X serves these purposes and more. It may take a little longer to learn than other types of DOS for the 8-bit ATARI 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 SpartaDOS operations in a tutorial format. Please read trough the chapter and perform the examples. Do take time to experiment. This will get you well on the way to becoming a SpartaDOS power user.

The Command Processor
SpartaDOS differs from ATARI DOS 2 types in many ways, but the most apparent is the user interface, the way in which you communicate with DOS and DOS communicates with you. Atari DOS and most others are menu driven; all options available are displayed on the screen and may be selected with a single key stroke. SpartaDOS uses a command processor (CP); at a prompt, the full command is typed. Those of you familiar with any DOS on personal computers, CP/M, UNIX, and many operating systems on other computers will recognize the CP interface. Each of these user interfaces have strong and weak points, but most users who take the time to become familiar with the operation of SpartaDOS find its CP interface to be very powerful and flexible. A menu is also included to make file maintenance very quick and easy. This is especially true with multiple files.

Getting Started
To get started, insert the SpartaDOS X cartridge into the cartridge slot (the left slot on an ATARI 800) and turn the computer on. If you happen to use a different host than a cartridge for your SpartaDOS X, please attach it to your system as explained in the respective manual and turn it on.

Using a 400 or 800 with 48 KB or any XL/XE computer with 64 KB of RAM, you will see a message like this:

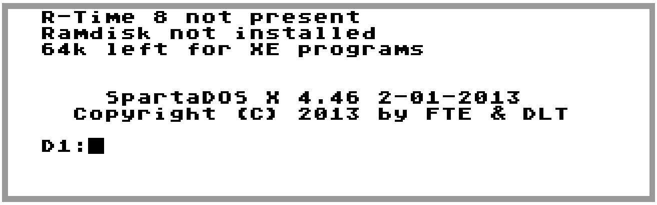


```
R-Time 8 not present
Ramdisk not installed
No extended RAM

    SpartaDOS X 4.46 2-01-2013
  Copyright (C) 2013 by FTE & DLT

D1:■
```

Fig. 1: Boot information - 48 or 64 KB memory

Information about the memory found by SpartaDOS X. The information line about the realtime clock R-Time 8 is of no importance yet.

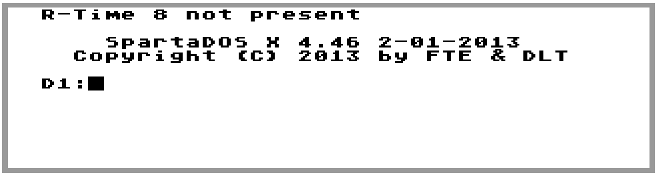
If your XL/XE computer comes with 128 KB of RAM, you will see this:



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


    SpartaDOS X 4.46 2-01-2013
  Copyright (C) 2013 by FTE & DLT

D1:■
```

Fig. 2: Boot information - 128 KB memory

If more than 128 KB of RAM on XL/XE or more than 48 KB on 400/800 (Axlon type) are detected, only the realtime clock and version information is displayed. This notes that SpartaDOS X has obviously found more than sufficient memory in your ATARI to create a convenient environment.



```
R-Time 8 not present

    SpartaDOS X 4.46 2-01-2013
  Copyright (C) 2013 by FTE & DLT

D1:■
```

Fig. 3: Boot information - sufficient memory found

SpartaDOS X has an intelligent memory management, which tests and configures the memory during boot up and is customizable.

The default setup configures memory found according to these schemes:

400/800
- 48 KB main memory
- 16 KB extended memory for SDX internal use
- The rest of the extended memory is used as ramdisk "O:".

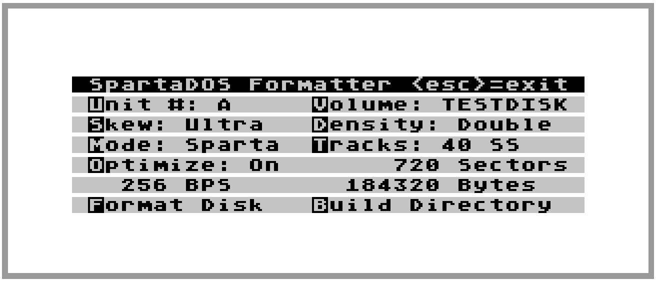
XL/XE
- 64 KB main memory
- 64 KB extended memory reserved for BASIC XE programs
- 16 KB extended memory for SDX internal use
- The rest of the extended memory is used as ramdisk "O:".

Type **CHKDSK O: /X** and press **RETURN** to see detailed information about your ramdisk.

Note: To restart SpartaDOS X type **COLD** and press **RETURN**. There is no need to switch your ATARI off and on again several times during this introduction except it might lock up.

Formatting a Disk

Before you start to explore the command processor, you need to have a floppy disk with which to experiment (or another medium, if you are familiar with it). Place a new disk in drive #1, type **FORMAT** and press the **RETURN** key. You do not need to type the D1:. Throughout this chapter, you will only need to type the text after the prompt. You should press RETURN after each line typed unless instructed otherwise.



```
 SpartaDOS Formatter <esc>=exit
Unit #: A       Volume: TESTDISK
Skew: Ultra     Density: Double
Mode: Sparta    Tracks: 40 SS
Optimize: On         720 Sectors
  256 BPS         184320 Bytes
Format Disk     Build Directory
```

Fig. 4: Formatter Menu

'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 ATARI 8-bit compliant drive with the respective drive software.

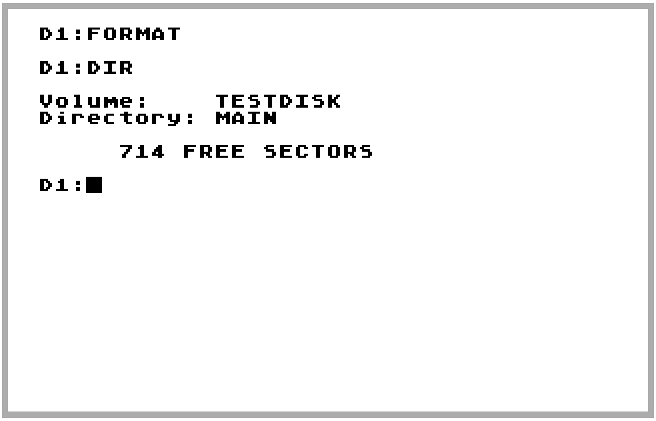
The blinking cursor in the unit field asks for input of the drive id. Press the **1** key to select drive #1 or any other valid drive number depending on your setup. Next the tool will present the facts it found about the drive like number of sectors, sector size and number of bytes. Those values will 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.

Now, press the **V** key, type **TESTDISK**, and press the **RETURN**. This sets the volume name of the disk to 'TESTDISK'. Make sure you have a new disk in drive #1, press the **F** key, and press the **RETURN** key to format the disk. 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' behind '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. In the standard setup it has the drive id 'O'. Experiment a bit, if you like.

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



```
D1:FORMAT

D1:DIR

Volume:     TESTDISK
Directory:  MAIN

     714 FREE SECTORS

D1:█
```

Fig. 5: First Formatted Disk

Disk Directory

After the format is completed, press the **ESC** key to return to the D1: prompt. Now type **DIR** and press the **RETURN** key. You will be prompted with the information about the just formatted medium. You may type **CHKDSK A: /X** and press **RETURN** to see more details about the just formatted medium in drive #1 (or A).

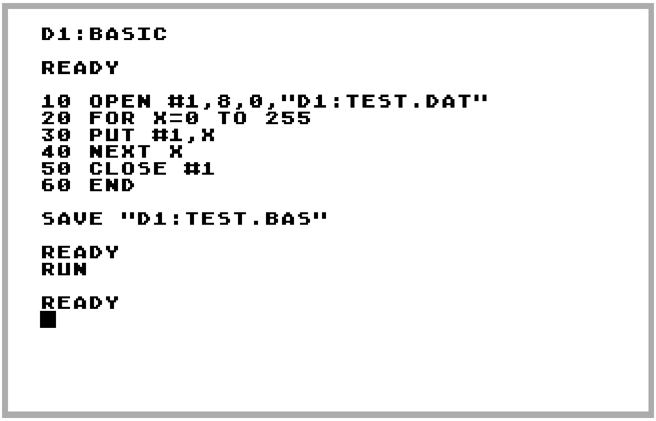
Creating Test Files

The disk has no files on it now. To be able to experiment with commands, you need some files on the disk. If you have a XL or XE computer (except 1200XL) type **BASIC** and press **RETURN**.

If your ATARI is a 400, 800 or 1200XL make sure that you have the ATARI BASIC cartridge installed in the top of the SpartaDOS X cartridge. If you do not, turn the computer off, insert the BASIC cartridge into the SpartaDOS X 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** and press **RETURN**.

If BASIC is available in one or the other way, regardless of the computer type, you should now see the BASIC prompt READY.

Type in the short BASIC program as depicted on the next screenshot, store it (SAVE) on disk and execute it (RUN).

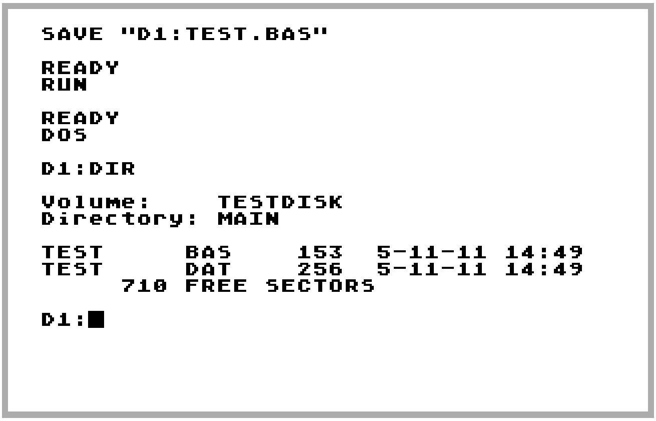


```
D1:BASIC

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: BASIC Program 'TEST.BAS'

Now type **DOS** and **RETURN** to get back to the DOS prompt.

Next type **DIR** and **RETURN**. You will now 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 SpartaDOS X is different from any other DOS. The first eight characters (four in this case) are the *filename.* The next three are the *extension.* The number to the right of the extension is the length of the file in *bytes,* not sectors. This is followed by the date and time. Your date, time, and free sector count may vary. 'TEST.BAS' is the BASIC program you just typed in, and 'TEST.DAT' is the data file it created. This format is called SpartaDOS File System (SDFS) and holds more information than any other on the ATARI 8-bit platform.

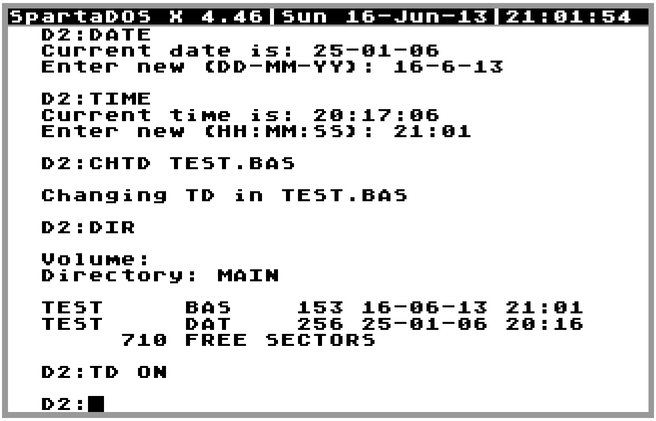
Setting the Time and Date

If you do not have any supported hardware clock, you will need to set time and date manually. This feature allows you to keep track of when your files were created and which is the latest version of a particular file. To do this, type **DATE** and **RETURN**. 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, and press the **RETURN** key. Next type **TIME** and **RETURN**. Again, the actual time and date displayed may vary from the examples. Now enter the new time and press RETURN. The time should be in the format 'HH:MM:SS' and based on a 24 hour clock (for example, 5:30 PM would be 17:30:00). Now time and date are set.

Now 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 has the current time and date.

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



```
SpartaDOS X 4.46|Sun 16-Jun-13|21:01:54
  D2:DATE
  Current date is: 25-01-06
  Enter new (DD-MM-YY): 16-6-13

  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 16-06-13 21:01
  TEST     DAT    256 25-01-06 20:16
       710 FREE SECTORS

  D2:TD ON

  D2:
```

Fig. 8: Time and date display line

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

If you always refer to the software clock of SpartaDOS X, it is helpful to adjust the AUTOEXEC.BAT file 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.

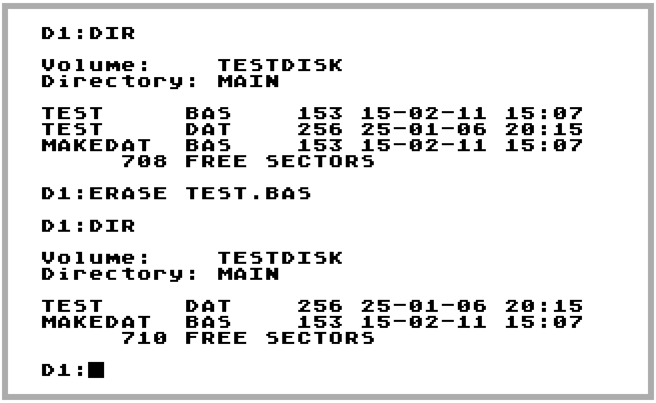
Parameters

Many commands require parameters. In the example for the command CHTD, the filename 'TEST.BAS' part 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 SpartaDOS X 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.

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 command (see chapter 4).



```
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

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.

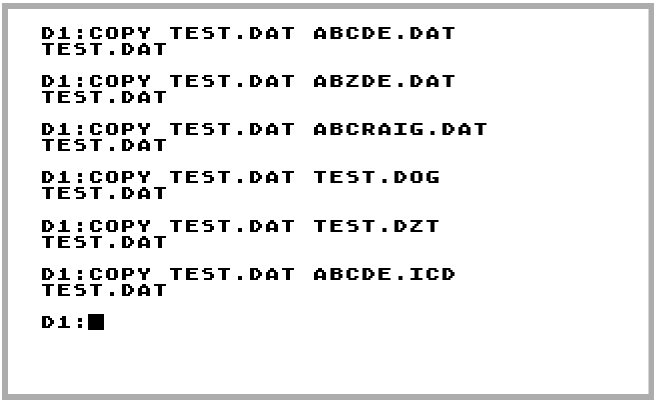
Type **ERASE TEST.BAS** and list the directory. The file TEST.BAS is gone.

Wildcards

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

There are two wildcards used in SpartaDOS 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.

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.

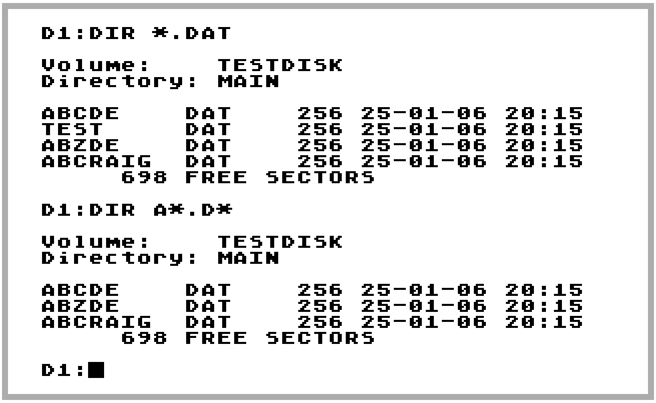
What this means is that you can add a file name and extension to the DIR command to get a partial listing from the directory.

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.



```
D1:DIR *.DAT

Volume:     TESTDISK
Directory:  MAIN

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
     698 FREE SECTORS

D1:DIR A*.D*

Volume:     TESTDISK
Directory:  MAIN

ABCDE     DAT     256 25-01-06  20:15
ABZDE     DAT     256 25-01-06  20:15
ABCRAIG   DAT     256 25-01-06  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.

Directories

What you have seen so far when listing the disk directory is the main directory (it even says so at the top). SpartaDOS 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 SpartaDOS) 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.

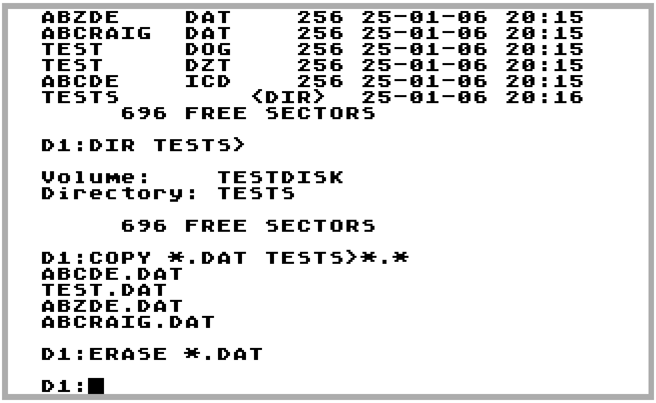
Subdirectories can be thought of as folders in the filing cabinet. If you took all of your picture files and put them in a folder labeled 'PICTURES', then you would only have to search the drawer until you found the 'PICTURES' folder, then search it for the desired picture file. Similarly, you could 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 **M**a**K**e **DIR**ectory 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>.

The '>' character preceding the name signifies it is a directory, not a file (see the previous section of wildcards). You should see an empty directory that looks just like an empty disk except that the name of the directory is now 'TESTS' instead of 'MAIN'. This shows you that it is the directory TESTS, not the main directory. The free sector count will also be less than that of an empty disk.



```
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 for copying

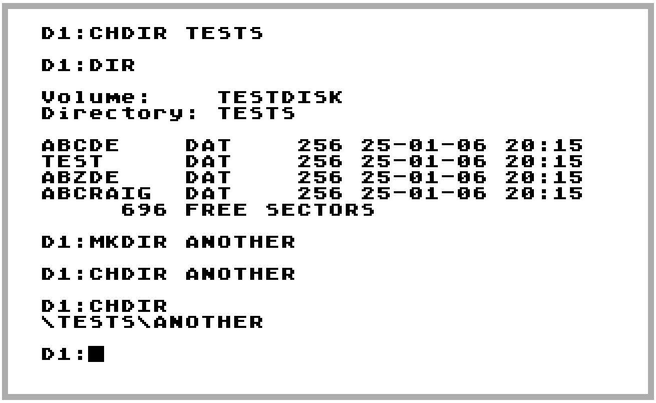
COPY is another command that allows wildcards. The next step is to copy all of the files ending in '.DAT' from the main directory to the subdirectory TESTS. And thereafter those files will be deleted from the main directory.

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".

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**.

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.



```
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

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 if you want to just get back one level towards the root directory from the directory you have entered (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 (such as fonts) are on a disk.

Running Programs
To run binary files from SpartaDOS 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, SpartaDOS would try to run a program named DEMO.COM.

BASIC,CAR, and X
As demonstrated earlier, to enter internal BASIC in the XL/XE or XEGS machine type **BASIC**. To enter an external cartridge (such as ACTION!, MAC/65, BASIC XL, BASIC XE, etc.) type **CAR**.

It is never necessary with SpartaDOS X to hold down the OPTION key to disable internal BASIC while booting or to remove the 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, "DiskRx", from the SpartaDOS Toolkit, will not run with any cartridges installed. To run it from SpartaDOS X, it would be necessary to type **X DISKRX** or **#DISKRX**.

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 (see Chapter 8 – Configuring Your System).

Hard disks users can configure DOS in a way that it automatically disables the ROM module while executing a program (see 'COMEXE.SYS' in chapter 8 for more details).

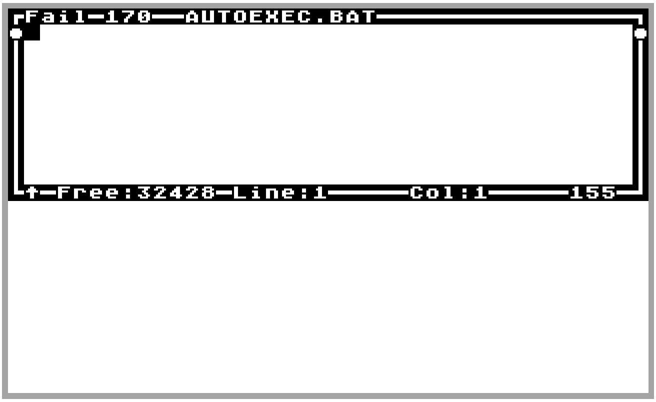
Building Batch Files

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, SpartaDOS X 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 realtime clock. Switch to the main directory of your boot drive. Now on the command line just type

ED AUTOEXEC.BAT

SpartaDOS X own editor starts and opens up the file AUTOEXEC.BAT. Since you create a new file there will be an empty screen. The fail message is of no importance yet.



```
Fail-170--AUTOEXEC.BAT

↑-Free:32428-Line:1     Col:1     155
```

Fig. 14: Use ED to create AUTOEXEC.BAT

Now type in the needed commands with a RETURN at the end of each line.

TD ON
TIME
DATE

Next press <ESC> once and <RETURN>. Your new AUTOXEC.BAT will be saved. Type COLD <RETURN> on the command line and watch the execution. Please look up the ED command in chapter 4 for more details.

Batch files are one of the most useful features of SpartaDOS X. 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.

Practice

Now that you have a basic understanding of the operation of SpartaDOS X 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 ATARI DOS you may feel somewhat restricted by the menu!

DOS 2 Equivalents

The following is a list of the commands from the ATARI DOS 2.0s menu and their equivalents in SpartaDOS X:

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

SpartaDOS X comes with numerous commands that have no equivalent in ATARI DOS 2.x. It also supports any drives that are legal in the ATARI 8-bit system, ramdisks, time/date stamping, subdirectories, and more.

3 SpartaDOS X Overview

This chapter is an overview of the filename and pathname conventions, device identifiers, and general usage. It is assumed in this chapter that you 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 SpartaDOS X and ATARI computers. SpartaDOS X is far more advanced than any other DOS for ATARI's 8-bit computers.

As a simple example, SpartaDOS X allows drive identifiers like 'A:' and pathnames like '>DOS>SUB2>MYPROG.BAS'. It supports most of the common hardware from nowadays as well as from the good old days of home computing.

Drives

Originally, most users of ATARI 8-bit 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 the manual 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 or report it in the corresponding thread at http://www.atariage.com/.

Filenames

The basic form of the filename is identical to SpartaDOS 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 upper case letters.

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

The underscore character ('_')

The underscore 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.

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 SpartaDOS
.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.

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.

Directories
The disk is broken up into directories, each of which may contain up to 1,423 entries (earlier versions of SpartaDOS have a limitation of 126 entries - in fact those versions will not read SpartaDOS X directories beyond the 126th entry!) The root directory is named 'MAIN' and other directories (which are called subdirectories) can be created under 'MAIN' (See the MKDIR command.). For directory names the same rules apply as for filenames; they may also have an extender.

Note: While SpartaDOS X 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 KB in size. Large directories will slow down disk access considerably, especially when opening new files.

When you display 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.

Pathnames
Since SpartaDOS can have more than one directory on each disk, it 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 SpartaDOS to begin at the root directory (MAIN). Also, if one or more '<' characters (or '..\' strings) begin a pathname, SpartaDOS 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 SpartaDOS X 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 the command line argument and prevents it from being interpreted as a redirect command.

For example,

```
        DIR <<PICTURES>
```

does not (as it did in previous versions of SpartaDOS) 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 SpartaDOS X 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.

Command Length

The maximum length of a line that will be accepted at the command line is 63 characters. There is no warning when this limit is exceeded. The additional characters will simply be ignored. This 63 character limit includes the command name itself but not the prompt.

Filespec

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

Device Identifiers

In earlier versions of SpartaDOS all device identifiers were based on the CIO device table. SpartaDOS X 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 SpartaDOS X "kernel", it is referred to as 'CON:'.

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

- SpartaDOS X was designed to feel exactly like an MSDOS 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 afore mentioned are just a few.

So, here is what we have come up with. Using the command processor, the devices are as follows:

SpartaDOS X can handle up to fifteen drives attached simultaneously to the computer. At the SIO level the drives are numbered from 1 to 15 ($01-$0F). Drives 1-9 have, just as in SpartaDOS X versions before 4.40, identifiers 1: ... 9: or A: ... I: in the Command Processor, and D1: ... D9: or DA: ... DI: in BASIC.

The drives 10-15 have letter identifiers only: J: ... O: in the Command Processor, and DJ: ... 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.)

DSKx: "DSK" is the official device identifier for your drives - since it is assumed, you need never type it.

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.

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 SpartaDOS versions, this was called 'E:')

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

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

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.

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 SpartaDOS X "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 SpartaDOS X "kernel" devices, but through the CIO you may access both sets of devices.

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 these directories, respectively).

Volume Names

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

- To better organize your media by naming them.

- For SpartaDOS to quickly tell the difference between e.g. the current diskette and the next diskette you put in the drive (since there is no way to tell when the door opens on the drive).

If you display a directory of a medium in an ATARI DOS 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 ATARI DOS formatted media, the buffer system does not remember anything about the last access to these media. Therefore, ATARI DOS formatted media are much less efficient and slower. In addition, loading binary files, especially those with many segments, can take *considerably* longer from an ATARI DOS medium than from a SpartaDOS medium. It is recommended that files be copied to SpartaDOS media or ramdisks before running.

Disk Format Compatibility

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

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

SpartaDOS 2.x and 3.x will flawlessly read from or write to media formatted and/or written to by SpartaDOS X. But there are exceptions: 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. DD512 formatted media are not compatible with any other SpartaDOS before X Version 4.4x.

SpartaDOS 2.x and 3.x will ignore the new file attributes (Archived and Hidden). They will not acknowledge these, nor will they change them.

Using External Cartridges with SpartaDOS X

SpartaDOS X is in a "piggy back" cartridge, meaning that an external cartridge may be plugged into the top of the SpartaDOS X cartridge. If you are using an 800 computer, SpartaDOS X should be plugged into the left cartridge slot in the computer. If you have a 130XE connected to a MIO, you may plug the SpartaDOS X 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 SpartaDOS X into it. You can plug it into the SpartaDOS X 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 SpartaDOS X cartridge into the R-Time 8 or plug the R-Time 8 into the SpartaDOS X cartridge.

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

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

Note: Some hosting platforms for SpartaDOS X do not provide an option to plug in a cartridge. Newer system like e.g. the TurboFreezer and the emulator Altirra do have a provision for it.

4 The Command Processor – Commands

The description of the command processor is broken down into two chapters. The first ("The Command Processor - Commands") is a listing of SpartaDOS X 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 more detailed information on some of its 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.

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 SpartaDOS X remains compatible with older SpartaDOS versions. Additionally, we try and maintain command similarities for people who are familiar with MSDOS.

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:' directory or may reference one of those files. Most of the SpartaDOS X cartridge content is devoted to these external commands.

"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 SpartaDOS X 4.4x on this command has been made available. Since there are several versions of SpartaDOS X - old and new ones – it helps to distinguish them. Commands without this remark were already distributed in before with the ICD version of SpartaDOS X.

"Remarks" include all the details and special rules of command usage. The remarks may also show usage examples. A good way to learn SpartaDOS X 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 SpartaDOS X and enables you to become a SpartaDOS X 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 SpartaDOS, 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 SpartaDOS X there are help files in form of man pages available to be handled by the 'MAN' command. Many platforms for SpartaDOS X 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 SpartaDOS X Toolkit.

The SpartaDOS X Toolkit enhances the SpartaDOS X system by more external commands, 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 . . .

APPEND Command

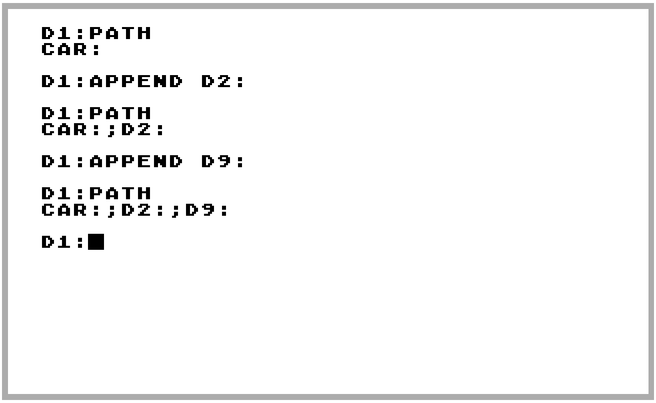
Purpose
Append the given path at the end of the $PATH variable.

Syntax
APPEND *pathname*

Type
External - on device CAR:

Availability
As of SpartaDOS X 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 directory to the $PATH without manually rewriting all the paths that $PATH contains.



```
D1:PATH
CAR:

D1:APPEND D2:

D1:PATH
CAR:;D2:

D1:APPEND D9:

D1:PATH
CAR:;D2:;D9:

D1:█
```

Fig. 15: Append a temporary path

ARC (Archive Files) Command

Purpose

Create and maintain file archives.

Syntax

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

Type

External - on device CAR:

Remarks

ARC is based on and compatible with ARC.EXE by System Enhancement Associates. ARC will take a group of files and 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 the syntax, command list, and options. 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. Update will look at the date of the files in the archive, replacing files with a newer date, and add all files (from file list) which do not currently exist in the archive.

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

D Delete file(s) from the archive. Delete 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 from an archive. 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 AH DOCS B:\READ\TXT\*.TXT F:\PRINT\*.DOC** will extract from the archive file 'DOCS.ARC' all TXT files and write them back to subdirectory 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:.

Note: ARC X still has some problems in packing long files (probably more than 64 KB) and sometimes crashes.

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 created of each file in the archive as well as the number of files and total size of files if extracted.

V Verbose list of file(s) in archive. This command 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 (64 columns or more → CON64|80.SYS), the complete listing is displayed.

E.g., **ARC V FOOBAR** lists all the contents of the file FOOBAR.ARC, whereas **ARC V FOOBAR A*.COM** lists COM files only whose names start with A.

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

Valid "**Options"** are:

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.

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

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 but still stays encrypted.

E.g., **ARC AHGICD STUFF DOIT.DOC DOIT.COM READ.ME** will add the 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 KB machines (USE OSRAM) and 180 files per archive on computers which use the extended memory mode (USE BANKED). ARC will not run on 48 KB 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. SpartaDOS X 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. The stowage methods used in ARC are as follows:

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.

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.

Note: 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.

ATR (Attributes) Command

Purpose

Sets/clears file attributes in the directory. Replaces the Protect and Unprotect functions from older SpartaDOS versions.

Syntax

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

Alias

ATTRIB

Type

Internal

Related

DIR

Remarks

SpartaDOS X adds two new attributes to the standard SpartaDOS directory entry - these are the hidden and archived bits. The old commands PROTECT and UNPROTECT were used to set or clear the protection bit. With SpartaDOS X, the ATR command replaces the old commands and works with the new attributes.

Although many other commands allow the usage of the 'S' (subdirectory) attribute, it is illegal to attempt to change this status bit as it would corrupt the subdirectory integrity. Therefore, ATR does not affect this.

Note that although the syntax of the ATR command looks similar to that of the DIRectory or TYPE commands, the attributes here are not the scan mode, but describe the set(+) / clear(-) attributes operation to be performed on the directory entry that matches the given filespec. This means that the scope of the ATR command is all files matching the filespec (including those files which are hidden).

The directory entry attributes are as follows:

A   Archived file. This attribute is cleared whenever a file is created or updated. The archive bit is set when the file is backed up by a program such as FlashBack!. This attribute is not related to the ARC command.

H   Hidden file. You may hide files and/or subdirectories. If a file is hidden, you may load it as a command only - commands such as TYPE and COPY will not see hidden files (unless you specify attributes with those commands). The file is hidden when this bit is set.

P Protected file. You may not ERASE, or update protected files. Use the ATR command to protect or unprotect files. The file is protected when this bit is set.

S Subdirectory. This attribute 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.

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 new attributes, please refer to the "Technical Information" chapter.

Notes: In all versions of SpartaDOS X the ATR Command works on directories as well. So be aware of your naming actions, since hidden directories will still be there.

Additionally, please be aware of the fact that the available attributes are not always fully recognized by all commands, to which they should apply. The concept for them within SpartaDOS X is still under development.

BASIC Command

Purpose
This command enters the ***internal*** BASIC in your XL/XE machine.

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

Type
External - uses CAR.COM on device CAR:

Related
CAR, SET

Remarks
Without a filename control is given to the ***internal*** BASIC of your XL/XE computer. 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. See Appendix E (→AUTORUN.SYS) for details.

To automatically load and run a BASIC program from the command processor, read about *I/O Redirection* in the "Advanced Features" chapter.

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.

SpartaDOS X has a MEM.SAV facility somewhat like that of ATARI DOS 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, the memory-save feature is disabled.

The $BASIC variable has no default value, unless RAMDISK.SYS is installed. If $BASIC has not been set by the user (SET command), 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 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
```

(See the SET command for a further explanation.)

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 the ESC key 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 (see the LOAD command). 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 (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 pages 4-6. This means that you may alternate between BASIC and CARtridge without losing what you were working on. When you enter BASIC the memory-save file is loaded, allowing you to edit a BASIC program, 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 SpartaDOS X cartridge and the external cartridge plugged into the SpartaDOS X cartridge, if any, to be disabled. This will have the same effect as typing COLD /N from the command processor.

BLOAD Command

Purpose
Loads the given file into the given memory area starting at the given address.

Syntax
BLOAD [d:][path]fname[.ext] $address
BLOAD [d:][path]fname[.ext] address

Type
External – on device CAR:

Related
LOAD

Availability
As of SpartaDOS X 4.40.

Remarks
The file *fname.ext* is loaded as a raw data block into the specified area, and then control is handed back to the Command Processor.

There are no checks done, whether the file fits in memory, or if vital operating system 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.

BOOT Command

Purpose

This command tells a SDFS-formatted disk/medium to boot a particular program at start up (usually a disk-based DOS).

Syntax

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

Type

Internal

Related

COLD, FORMAT

Remarks

The DOS loader on the first three sectors of each SpartaDOS formatted diskette (version 2 and above) can load and run files in the same manner as a command file. Normally DOS is loaded, but anything could be loaded as long as it avoids the loader memory ($2E00-$3180).

The FORMAT command does not put any DOS on the data medium it formats!

So, if you want to create a bootable DOS data medium, you must COPY a 'disk' based, compatible DOS (SpartaDOS 2.x or 3.x, BeweDOS, Micro-SpartaDOS or RealDOS) to the medium and use the BOOT command. If you just use SpartaDOS X, this will never be necessary since SpartaDOS X boots from cartridge. If you still use disk based versions of compatible DOS, you may wish to create bootable media. Of course you may still use the e.g. the XINIT command to format and install DOS on your media.

This command is the most misunderstood command in SpartaDOS, 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 diskette. (See "Technical Information")

- If the file which is set to boot is either ERASEd or COPYed over, the boot flag is cleared - you will get the message 'Error: No DOS' when attempting to boot that diskette until you set a new file to boot!

- The file you set to boot may reside anywhere on the medium - even in a subdirectory.

- This command does not work with SpartaDOS 1.1, since it does not contain a 3 sector booter! SpartaDOS 1.1 has a simplistic booting scheme which just loads a number of consecutive sectors.

Notes: The boot command is very handy if you like to install a disk based version of a SDFS compatible DOS on larger media like a hard drive. Format it using SDX, copy the respective DOS onto it and make it bootable with the BOOT Command.

Please be aware that the new density using 512-byte-sectors cannot be booted by the ATARI XL/XE operating system. Therefore settings to boot a program will not work on such formatted media.

CAR Command

Purpose

This command enters the cartridge plugged into the top of the SpartaDOS X cartridge. If a filename is specified, then that binary file is loaded and run with the cartridge enabled.

Syntax

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

Type

External - on device CAR:

Related

BASIC, COLD, SET

Remarks

If no filename is given, control will be handed over to the cartridge plugged into the SpartaDOS X cartridge. If a filename is specified, that binary file will be loaded and run with the cartridge enabled. 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 '/N' option returns to the cartridge after running fname, instead of to the command processor, which is the default. See Appendix E (→ AUTORUN.SYS) for details when using e.g. BASIC cartridges.

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.

SpartaDOS X has a MEM.SAV facility somewhat like ATARI DOS 2, but with much more power. The environment variable $CAR should be set to the file you wish to use as the memory-save file for the cartridge. If no such environment variable exists, the memory-save feature is disabled.

The $CAR variable has no default value, unless RAMDISK.SYS is installed. If $CAR has not been set by the user (SET command), 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'. To see the current value of $CAR (and all the other environment variables) just type:

```
SET
```

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

```
SET CAR
```

(See the SET command for a further explanation.)

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. 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 (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 the ESC key 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 (see the LOAD command). 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 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 (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 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 loosing 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 cause SpartaDOS X to be disabled, while leaving the external cartridge enabled. This is the same as typing COLD /C from the command processor.

CHDIR (Change Directory) Command

Purpose
This command changes the current (working) directory on the specified drive, or displays the current directory path if no path is given.

Syntax
CHDIR [d:][path]

Alias
CD & CWD

Type
Internal

Related
MKDIR, RMDIR, PATH

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 SpartaDOS looks to find files whose names were entered without specifying a directory. If you do not specify a drive, the default drive is assumed. If you enter the CHDIR command with no parameters, the current directory path of the current drive is displayed. This mode is the same as the ?DIR command from SpartaDOS 3.2.

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

This command has no effect on MyDOS diskettes 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 diskettes. SpartaDOS diskettes have a volume name, a random number, and a write count for disk change detection (see "Technical Information").

CHKDSK Command

Purpose
Show volume, free/total disk space, and sector size of the selected drive (or partition/medium).

Syntax
CHKDSK [d:] [/X|V]

Type
External - on device CAR:

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 ATARI DOS 2 style diskettes will appear as "DOS 2.0".

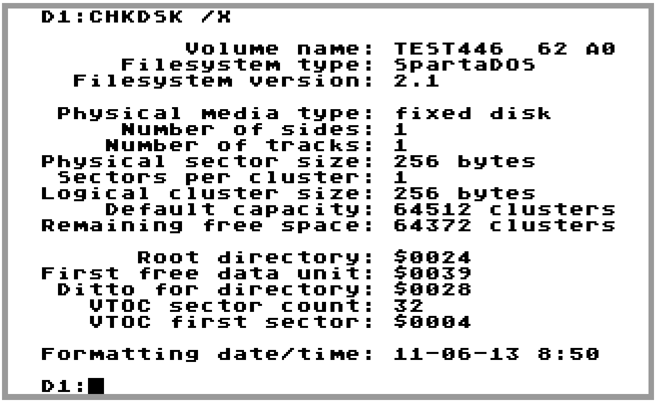
The '/X' option causes an extended disk information to be displayed. This way you can quickly review the details of how your medium is set up.

The '/V' option is a far more advanced tool:

- The disk bitmap (VTOC) will be loaded into memory and analyzed,
- The information about remaining free space is compared with the amount of free space indicated by the boot sector. This allows to check quickly, if there are lost sectors on the disk.
- The [/V] option will only work with regular SpartaDOS disks.

The disk write-lock status known from SpartaDOS versions before 4.x is omitted - this feature is no longer supported. We found this to be more of a hassle than it was worth, and it did not protect you from formatting the diskette. (The write-lock feature of the MIO still works and is totally independent - it is a far more secure write-lock.)

Next a sample output of the CHKDSK command using the /X option:



```
D1:CHKDSK /X

         Volume name: TEST446  62 A0
     Filesystem type: SpartaDOS
  Filesystem version: 2.1

 Physical media type: fixed disk
     Number of sides: 1
    Number of tracks: 1
Physical sector size: 256 bytes
 Sectors per cluster: 1
Logical cluster size: 256 bytes
    Default capacity: 64512 clusters
Remaining free space: 64372 clusters

      Root directory: $0024
First free data unit: $0039
 Ditto for directory: $0028
   VTOC sector count: 32
   VTOC first sector: $0004

Formatting date/time: 11-06-13 8:50

D1:
```

Fig. 16: Check disk (or medium)

The two numbers following the volume name are used for disk change detection in cases where volume names are the same on both diskettes. 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.

CHTD (Change Time/Date Stamp) Command

Purpose

This command changes the time/date stamp on all files matching the given filespec to the current time and date.

Syntax

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

Type

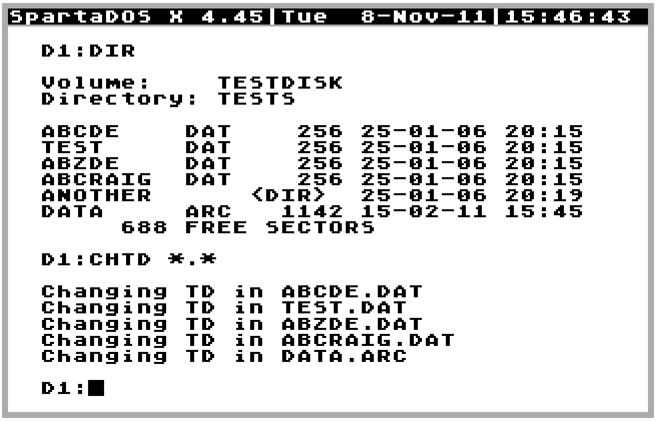
External - on device CAR:

Related

DATE, TD, TIME

Remarks

By default, this command will only change the time/date stamp on non-hidden and non-protected files - this may be overridden. (See ATR command for more information on attributes.) You must enter a filespec 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

CHVOL (Change Volume Name) Command

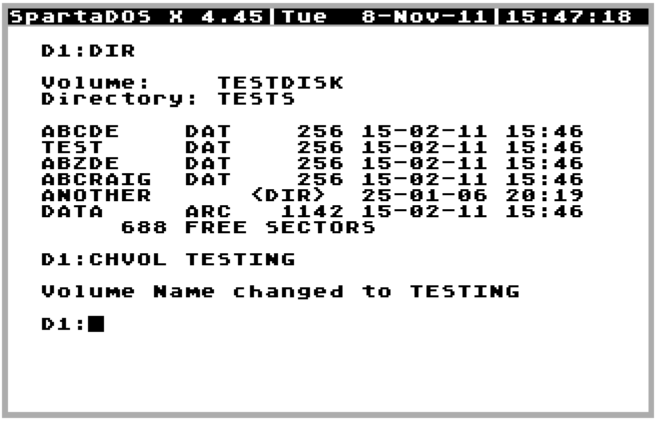
Purpose
This command changes the volume name on the specified drive.

Syntax
CHVOL [d:]volname

Type
External - on device CAR:

Related
CHKDSK, FORMAT, DIR

Remarks
This command will not change the volume name on ATARI DOS 2 diskettes since they physically have no volume name. Up to eight characters are allowed on SpartaDOS 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

CLR Command

Purpose
To delete unused system variables.

Syntax
CLR

Type
Internal

Availability
As of SpartaDOS X 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 the RESET key. 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".

CLS Command

Purpose
To 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 settings in your AUTOEXEC.BAT will be kept valid.

COLD Command

Purpose
This command reboots 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. (See the RAMDISK.SYS driver description.) This is equivalent to the SpartaDOS 3.2 command:

```
RUN E477
```

This command has two options:

C Reboot the computer with SpartaDOS X disabled and the cartridge plugged into SpartaDOS X 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 SpartaDOS X has been disabled, it will be necessary to turn the computer off and back on to re-enable SpartaDOS X.

Practically, this hardware torture may be omitted by booting INIDOS.SYS available from the SpartaDOS X Toolkit.

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

COMMAND (The Command Processor)

Purpose

This program allows you to enter commands and run other programs. It is not entered as a command itself but is automatically invoked when you enter DOS.

Type

External - on device CAR:

Related

All Commands

Remarks

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 (see the PATH command).

You will notice that the command processor itself is "external". This is to give you more memory (3-4 KB) 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.

COMP Command

Purpose

Compares the given files.

Syntax

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

Type

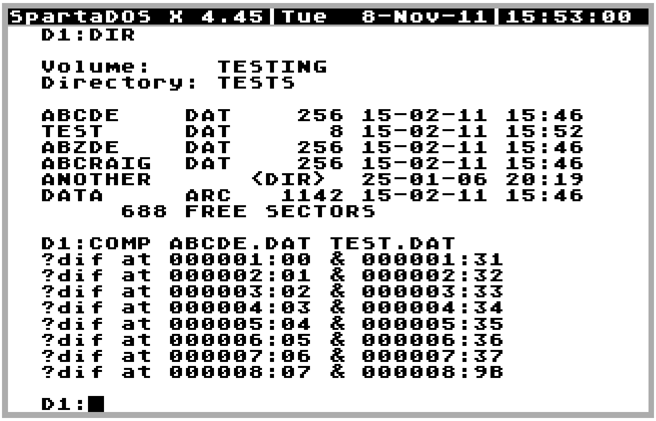
External - on device CAR:

Availability

As of SpartaDOS X 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      <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 offsets are starting positions, where the respective files get compared from. If none is provided, for both files 0 will be assumed.

CON (Console Modes)

Purpose
CON: drivers control.

Syntax
CON 40|64|80

Type
External - on device CAR:

Availability
As of SpartaDOS X 4.42.

Remarks
This command enables and disables the 64- and 80-column text modes handled by the CON64.SYS (see Chapter 8) and CON80.SYS drivers (see SpartaDOS X Toolkit). The commands 'CON 64' and 'CON 80' will try to enable the 64- and 80-column mode respectively - the respective driver must be loaded first for this action to succeed. 'CON 40' disables either mode switching the screen to the standard, 40-column text console. The message 'Mode not changed' means, that the respective driver was not loaded to the memory, or the screen is already in the requested mode.

Notes: In CONFIG.SYS the DEVICE QUICKED.SYS must precede DEVICE CON64 and/or DEVICE CON80 to get them working properly. DEVICE RAMDISK should be placed behind those in CONFIG.SYS, otherwise you will waste one bank of extended memory.

COPY Command

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.

To COPY files from one diskette to another having just one drive, the file either has to be COPYed from the source diskette to a ramdisk and then from the ramdisk to the destination diskette, 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 KB (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 following 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 (ANTIC DMA) 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:** will do.

The second command will send the content of the file to the printer.

The COPY command may also be engaged to append files by using a '/A' immediately following (no space) the destination filespec. (SpartaDOS 3.2 allows a '/A' when SAVEing any file to force append mode - SpartaDOS X 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 SpartaDOS X. 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.

DATE Command

Purpose
This command displays the current date and allows you to set the date.

Syntax
DATE [/T|dd-mm-yy or mm-dd-yy]

Type
External - on device CAR:

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):
```

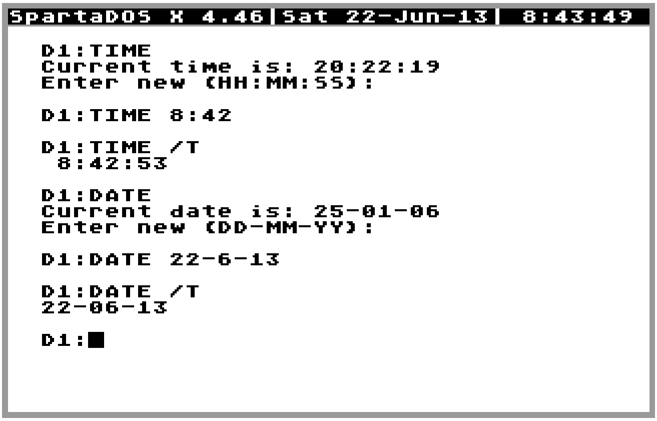
Enter the new date or just press RETURN to keep the current setting. The date format - 'DD' for day, 'MM' for month, and 'YY' for year – is obligatory to change the settings. Type 'DD' to change the day, 'DD-MM' to set day and month, all to change the year too. The space key is a legal delimiter.

When used with /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 settings apply here.

Without a proper clock driver installed this command will produce meaningless results. SpartaDOS X currently provides several drivers for real time clocks and for the ATARI software clock (→ JIFFY.SYS). By default, one of these drivers will be installed during boot up, but can be overridden by creating a custom 'CONFIG.SYS' file to change the preset. See chapter 8 (→ time keeping drivers) for more details.

Note: Sometimes a realtime clock 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 from the SpartaDOS X toolkit.



```
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

DELTREE Command

Purpose
Delete subdirectory trees recursively.

Syntax
DELTREE [/YV] [d:] path

Type
External - on device CAR:

Related
RMDIR

Availability
As of SpartaDOS X 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 SpartaDOS X 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' from the SpartaDOS X Toolkit to verify the structure of the file system and fix it. See more in Appendix E, "Using CleanUp with SpartaDOS X".

DEV Command

Purpose
Display the list of available/installed kernel devices.

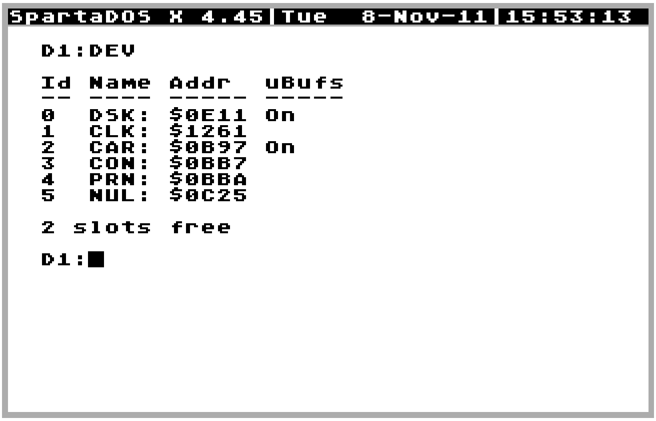
Syntax
DEV

Type
External - on device CAR:

Related
CHKDSK

Availability
As of SpartaDOS X 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

DF Command

Purpose
Display summary information about free space on all disks.

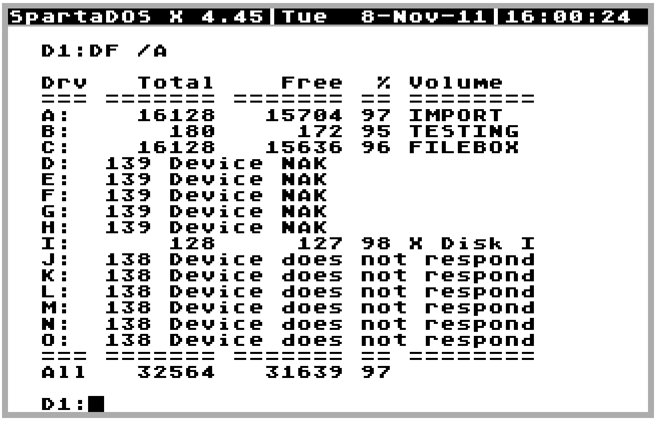
Syntax
DF [/A]

Type
External - on device CAR:

Related
CHKDSK

Availability
As of SpartaDOS X 4.40.

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



```
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.

DIR (Directory) Command & DIRS (Short Directory) Command

Purpose

Lists either all the directory entries, or only those matching a specified filespec. DIR will optionally give you a count of files listed.

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|W]

Type

Internal

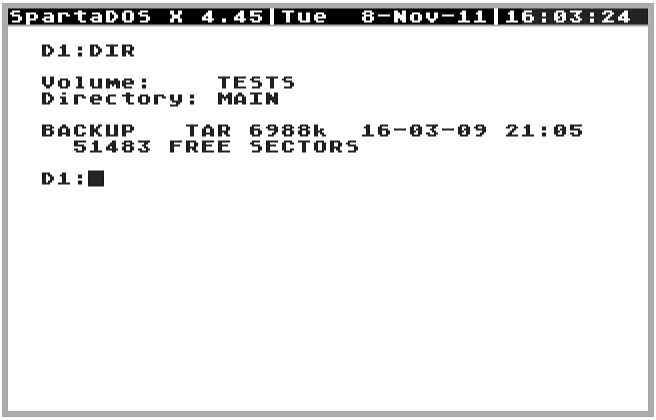
Related

ATR, FIND, MENU, PATH, PAUSE, PROMPT

Remarks

SpartaDOS versions before 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 SpartaDOS X 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.

SpartaDOS X 4.4x solves this problem: when a file exceeds 999,999 bytes in size, it is displayed in kilobytes, using the "k" character as indicator.



```
SpartaDOS X 4.45|Tue  8-Nov-11|16:03:24

  D1:DIR

  Volume:    TESTS
  Directory: MAIN

  BACKUP   TAR 6988k  16-03-09 21:05
    51483 FREE SECTORS

  D1:
```

Fig. 23: Display the directory

Short directory listings obtained by 'DIRS' contain subdirectory extensions (instead of 'DIR'). A colon printed in front of the file name is used to indicate the directory, like in MyDOS.

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

DIR displays the SpartaDOS file directory showing filename, extension, file size in bytes, date, and time created. It also shows a <DIR> in the size field when it sees a subdirectory, displays the Volume and Directory name at the top of the listing, and shows the Free Sectors count at the end of the listing. If you include a '/P' parameter, the DIR command will wait for a key press after displaying each directory screen (23 lines). The '/C' parameter will give a count of the number of entries displayed in that directory. '/W' lists the directory as file names only (without sizes, time stamps etc.) in as many columns as it can fit on the screen (i.e. in 64-column mode, for example, there are more columns listed than in GRAPHICS 0 with its 40 column size).

When reading an ATARI DOS 2 type diskette, the date and time are omitted for obvious reasons and the file size is roughly converted to bytes using a multiply of 125 (SD/ED) or 253 (DD) respectively (ATARI DOS 2 and clones use sector lengths instead of bytes in the directories so this can not be an exact file size representation.) All ATARI DOS 2 type diskettes will have a volume name of "DOS 2.0" and a directory name of "ROOT".

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). If you wish to see all files (including hidden files), simply enter **DIR +**.

Using a '+' with no attribute listed will match all files, regardless of attribute. This will work with any command that allows attribute selection.

The DIRS command has exactly the same syntax, but it displays the directory in Atari DOS 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 filename.

The attributes are:

A Archived file. This attribute is cleared (-) whenever a file is created or updated. Using a program like Flashback! will set this attribute.

H Hidden file. You may hide files and/or subdirectories. If a file is hidden, you may load it as a command only. Commands such as TYPE and COPY will not recognize hidden files (unless you specify attributes when using those commands).

P Protected file. You may not ERASE or update protected files. Use the ATR command to change this status.

S Subdirectory. This attribute cannot be changed.

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

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

Note: A subdirectory name has to be followed by a '>' or '\' character, if you wish to see the content of that directory.

DUMP (Display File in HEX Format) Command

Purpose
This command displays a file in HEX and ATASCII form.

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

Type
External - on device CAR:

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 (semi-graphics, 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 "DiskRx" from SpartaDOS Toolkit or "Eddy" from the SpartaDOS X Toolkit.

ECHO Command

Purpose

Enable or disable the "echo" in the Command Processor.

Syntax

ECHO ON|OFF

Type

Internal – executed by COMMAND.COM

Availability

As of SpartaDOS X 4.40.

Remarks

ECHO OFF disables echoing user commands, passed to the Command Processor from the command line or fetched from a batch file. ECHO ON restores the default.

The ECHO command as well "echoes" the text given as a parameter. ECHO TEXT simply displays the given text. To display the value of an environment variable, its name should be preceded with '$'-sign, for example:

```
ECHO $PATH
```

The preset is ECHO OFF.

Chapter 5 contains more information on this topic.

ED Command

Purpose

Enable text editor.

Syntax

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

Type

External - on device CAR:

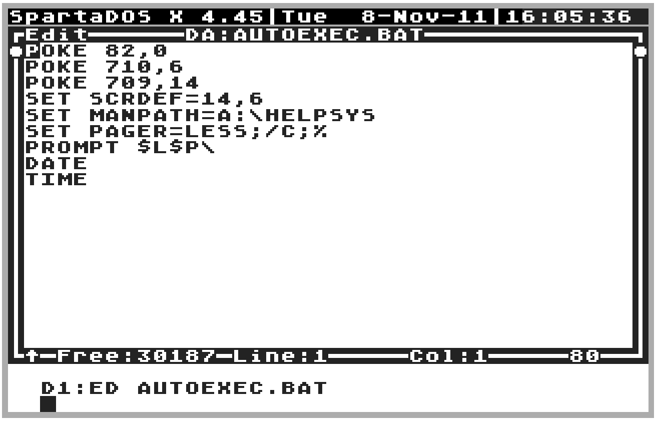
Availability

As of SpartaDOS X 4.40.

Remarks

ED.COM is a SpartaDOS X-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 KB, 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).



```
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. 24: Small text editor

Calling ED with a filename on the command line causes an attempt to load it. 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 Insert a new line before the tagged one (see Ctrl/T). The cursor moves to the next line.
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.

Notes: 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!

ERASE Command

Purpose

This command 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.

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, SpartaDOS X will prompt you with:

```
Erase ALL: Are you sure?
```

With SpartaDOS X 4.4x additional precautionary checks have been implemented to make sure that inputs like '?*.?*' will not accidentally kill your data. Use MENU for tagging files to delete.

FIND (Find Files) Command

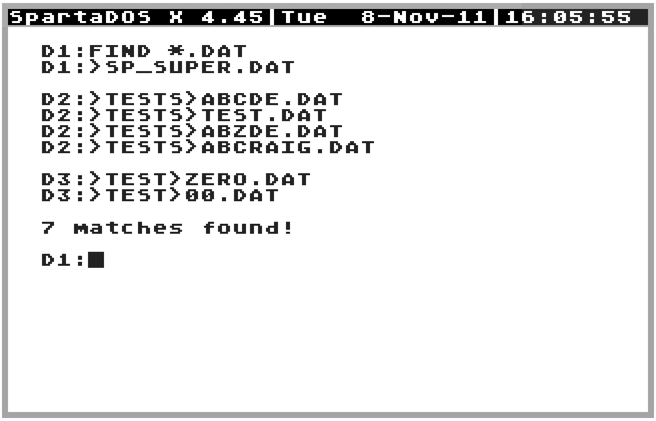
Purpose
This command searches specified areas in the system for files.

Syntax
FIND [d:]fname[.ext] or FIND device fname[.ext]

Type
External - on device CAR:

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. 25: 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 FTE's SpartaDOS Toolkit.

FMT (Format Text) Command

Purpose

Simple text formatter.

Syntax

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

Type

External - on device CAR:

Related

MAN, MORE, LESS, TYPE

Availability

As of SpartaDOS X 4.42.

Remarks

This is 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 it contains an EOL character 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). In other words, the text should not have been broken into lines. 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 – e.g. "First XLEnt Wordprocessor".

FMT is a filter command that can receive data from a pipe. 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.

Chapter 5 contains more information on pipes.

FORMAT Command

Purpose

Initializes a disk in either SpartaDOS or ATARI DOS 2 format. You may select density, sector skew, tracks, and volume name before formatting. It supports most known hardware configurations for your computer.

Syntax

FORMAT

Type

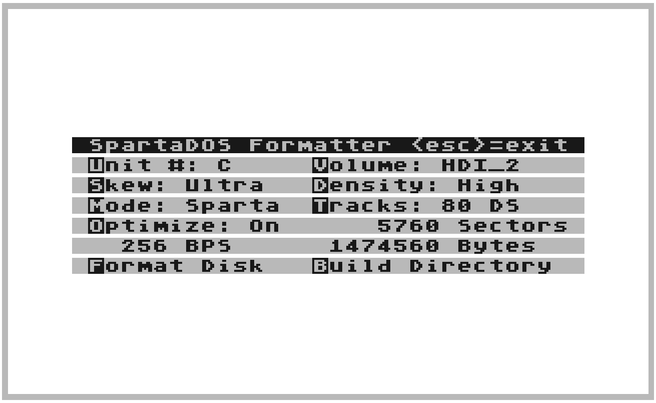
Internal

Related

BOOT, CHVOL

Remarks

The FORMAT command is a menu driven program, that allows you to initialize just about any type of medium that works with an ATARI 8-bit computer. It can be called from the command processor by typing FORMAT or from within a program with XIO 254 (see the "Programming with SpartaDOS X" chapter). This allows FORMAT to be used with most programs that support disk initialization.



```
SpartaDOS Formatter <esc>=exit
Unit #: C       Volume: HDI_2
Skew: Ultra     Density: High
Mode: Sparta    Tracks: 80 DS
Optimize: On       5760 Sectors
  256 BPS       1474560 Bytes
Format Disk     Build Directory
```

Fig. 26: Formatter menu

The formatter has been enhanced to support 1.44 MB floppy disk drives and larger file systems (up to 32 MB per partition is now possible), to handle larger sectors (up to 512 bytes), and to be able to use more drives. Apart from the three old ones being Single, Dual and Double, the density selection offers two new densities named DD 512 and, brand new, a density named High.

DD512 is the double density at 512 bytes per sector. The DD 512 density is usable with certain types of hard drive interfaces (KMK/JŻ IDE), and with TOMS floppy disk drives, either the original ones (TOMS 710, TOMS 720), or third party drives with TOMS Turbo Drive or TOMS Multi Drive extensions installed.

High formats $3^1/_2$ inch HD floppy disks in 1.44 MB. Following the PERCOM standard it can be used on devices being capable of handling this format like e.g. the 'HDI', a 'High Density Interface' for using standard $3^1/_2$ inch HD floppy disk drives.

Formatting a floppy diskette consists of several steps. At first the sector structure is written to the diskette. Now the DOS has a place to put the program information to. Next the directory structure is written to the diskette, wherein the DOS keeps track of the sector usage. Also ramdisks or partitions on hard disk drives can be initialized using the FORMAT menu, but only with the BUILD DIRECTORY option.

You can exit or quit the FORMAT menu at any time before formatting begins by pressing ESC.

After entering the FORMAT menu you choose the following parameters:

U **Unit** is the initial selection that must be made. The formatter needs to know 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. FORMAT 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, SpartaDOS X 4.4x attempts to read the existing volume name and presents it in the formatter menu as the default one.

Note: FORMAT will only write directories to ramdisks and hard drives (they won't be formatted). An internal ramdisk must be installed with the RAMDISK.SYS driver. A hard drive partition must be prepared with a program that should have been provided with the hardware.

O **Optimize**. SpartaDOS versions before 4.4x build directories in such a way that the last sector on the disk is marked as occupied and left unused. When the Optimize option is enabled in the formatter, that sector is reclaimed and assigned to the data area, so that you have one more free sector, than you usually have on freshly formatted media.

Note: There are some ancient hard drives, which 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 0 sector is not accessible, but it does count into the summary of existing sectors returned to the computer by the disk controller. This problem occurs e.g. in Supra Corp. and K-Products drives. On such a drive, the sector "reclaimed" by the formatter’s 'Optimize' function does not really exist. If you have such a drive, you should keep the ''Optimize'' off when building directories.

S **Skew** refers to the order in which the sectors are arranged in a given track and of course applies to real floppy disk drives only. The three valid choices are: Ultra Speed, High Speed and Standard. High Speed will automatically put the correct Ultra Speed skew on a disk using SpartaDOS 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. If you do select High Speed skew and the drive does not support a high speed mode, the format program will receive an error from the drive and then try to format under 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. Experiment a bit to find the best selection for your drives.

Note: The optimum skew will position the sectors so that after sector 1 is read and the drive CPU is ready for the next, sector 2 is directly under the head for reading; after 2 is read, 3 is directly under the head, etc. (Usually 2-8 sectors have passed under the head before the next sector can be read. This varies with drive speed and SIO baud rate.) If the skew is off, it may take a full disk revolution to read or write the next sector each time. No harm is done, the drive just reads and writes slowly.

M **Mode** is either **Sparta** for the SpartaDOS disk directory structure or **Atari** for all the ATARI DOS 2.0 clones and their directory structures in single and double density only. If you switch from Sparta to ATARI mode, impossible settings for formatting will automatically be re-adjusted to legal ATARI settings.

Note: FORMAT does not write any "DOS" files to a medium. For creating a bootable SpartaDOS disk, a "DOS" file from the SpartaDOS Construction Set must be copied the your formatted medium and then made bootable by the BOOT command. (See the BOOT command.) To create a bootable ATARI DOS diskette, ATARI DOS has to be engaged.

V **Volume** is a way of naming the media for organizational purposes. Up to eight characters are allowed on SpartaDOS formatted media. The volume name may contain any ATASCII characters except spaces. Volume is used on SpartaDOS media only and is not applicable to other DOS types.

D **Density** may be one of five types used with 8-bit ATARI computers. These are: Single, 128 bytes per sector FM, Dual, 128 bytes per sector MFM, Double, 256 bytes per sector MFM, DD 512, 512 bytes per sector MFM (FM and MFM refer to bit density where MFM writes twice the number of bits in the same area as FM), and high density enabling to use 1.44 MB floppy disk drives. Stock ATARI 810s only support Single, upgraded 810s (e.g. Happy Enhancement) double density. Stock ATARI 1050s support Single and Dual. ATARI 1050s with the US Doubler (or other enhancements), ATARI XF551s, and Indus GTs support single, dual, and double density. Most other disk drives for the 8-bit ATARI support Single and Double density. If 512 byte per sector are selected as density, SpartaDOS X 4.4x forces the Sparta mode as there is no possibility to build an ATARI DOS file system for the disk with 512-byte sectors. High density is supported e.g. by the HDI.

Note: Dual (or enhanced) density is now available for 1050 drives. SDFS format only.

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 diskette) and DS means Double Sided (2 heads with each writing on opposite sides of the diskette). All ATARI brand drives will use 40 SS except for the XF551 which is capable of 40 DS. Most of the other 5$^1/_4$ inch drives will be either 40 SS or 40 DS. (See your drive manual if you are not sure.) 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$^1/_2$ inch drives and high capacity 5$^1/_4$ inch drives with a similar interface. All drives with two heads will also format in the SS mode.

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

F **Format Disk** will start the physical format of a floppy diskette assuming you have entered all the other required parameters. It also writes the directory structure selected in Mode after the physical format and verify its completion. (The physical format and verify are functions of the floppy disk drive controller and not affected by the SpartaDOS VERIFY command.) CAUTION: The Format Disk procedure obviously destroys all previous information stored on the diskette.

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 fresh SpartaDOS directory structure to the drive unit selected, which means it will destroy all previous information stored in the ramdisk or hard disk partition.

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 FORMAT menu. The physical format of the ramdisk is provided by the ramdisk handler at installation.

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

As of SpartaDOS X 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 diskette must be formatted first, and then it is possible to check, if the diskette's capacity is the same as expected.

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

Indus GT notes: It should be mentioned here that the Indus GT before version 1.20 has a few known quirks or bugs. Because of this, Dual (Enhanced) Density is not supported on this drive for the ATARI file system, but it will work with the SpartaDOS file system. The Indus GT has also been known to keep spinning indefinitely when used in a system with US

Doubler enhanced 1050s. If you experience this problem, the best solution is to stop using mixed drives in your system.

CAUTION: The FORMAT command 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 1050 Turbo, Top Drive 1050, TOMS Turbo Drive or any other using the same method. Such a drive can be used with SpartaDOS X, 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 SDX Toolkit 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 only displayed 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 the drive.

KEY (Keyboard Buffer) Command

Purpose

This command installs a 32 character keyboard buffer and also links an "internal" KEY command into your system (for turning the buffer on and off).

Syntax

KEY ON|OFF

Type

External - on device CAR:

Remarks

The first time you use this command, it installs a keyboard driver into your system. The keyboard buffer will provide a faster key repeat and allow you to type ahead while the system is busy. Then the ON/OFF parameter is interpreted, enabling or disabling the keyboard buffer accordingly. Keys coming from the auto-repeat of the keyboard will not be buffered.

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.

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

LESS Command

Purpose
Paging text viewer.

Syntax
LESS [/C] fname[.ext]
LESS [/C] <<fname[.ext]

Type
External - on device CAR:

Related
MORE, TYPE

Availability
As of SpartaDOS X 4.42.

Remarks
This command is a little bit better version of MORE. If the text is shorter than 24 lines, and it fits entirely on the screen, the viewer behaves exactly (well, 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 or Return: scroll one line down
- Up arrow: scroll one line up
- Right arrow or 'F' or Space: scroll one page down (forwards)
- Left arrow 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

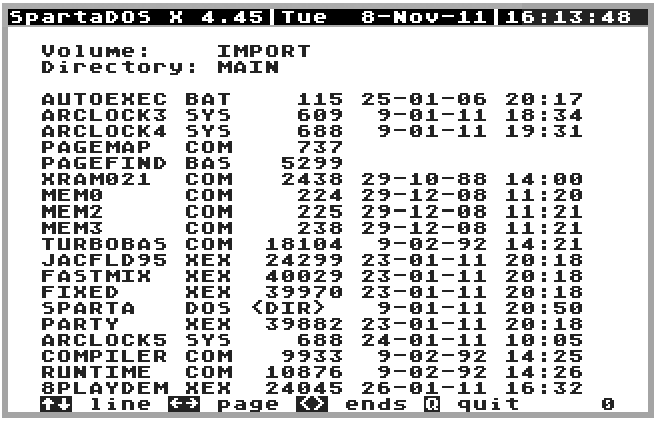
Some of them are shown in the bottom of the screen. The number in the bottom-right corner is the number of the text line displayed at the top of the screen.

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, i. e.:

```
D1:DIR | LESS
```

Chapter 5 contains more information about pipes.

LESS loads the entire text to the memory, so it cannot be used to view files, which are longer than its buffer (~ 35 KB, depending on the configuration).



```
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 XEX   24299 23-01-11 20:18
  FASTMIX  XEX   40029 23-01-11 20:18
  FIXED    XEX   39970 23-01-11 20:18
  SPARTA   DOS  <DIR>   9-01-11 20:50
  PARTY    XEX   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
  ↑↓ line ←→ page <> ends Q quit      0
```

Fig. 27: Less used to format a directory output

LESS now automatically detects MS-DOS (CP/M) and Unix line endings and converts the text accordingly on the fly. Therefore, you now may use the command to convert MS-DOS and Unix 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. Only EOL and TAB characters will be converted, but that's enough most of the time.

If LESS is invoked without parameters, it will expect input from CON: and therefore act like a memopad. Terminate it with <CTRL><3>; <BRK> or <RESET> abort it.

LOAD Command

Purpose
Loads a file (no run). This is useful for keeping commonly used commands resident in memory, thereby eliminating the need for these commands to load from disk. If no filename is used, all files previously loaded are removed from memory.

Syntax
LOAD [d:][path]fname[.ext]

Type
Internal

Related
MEM, SAVE

Remarks
If you LOAD a standard binary load file, the results are identical to those achieved with SpartaDOS 3.2. The file is loaded into memory and not run. There are a few primary uses:

- To load MAC/65 object files into memory and then SAVE them back as contiguous non-segmented binary files.
- To load a binary program prior to running a debugger (for testing purposes).

The only difference is that, if the program contains an INITAD segment, that will be executed.

One use of LOAD is to temporarily make external commands memory resident. This will only work with special SpartaDOS X relocatable external commands. See the MEM command for more details.

LOAD is used to:

- Keep an external command such as CAR or X or the command processor (COMMAND.COM) resident in memory.
- Remove all non-installed commands or programs from memory (use LOAD with no filename).
- Load a subprogram into memory for use by other commands.

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 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 read a short help screen. If MANPATH is not defined yet, you will see a respective notice at the end of the help screen.

MANPATH contains a list of directories to be searched for text files, in an identical manner as the variable PATH contains a list of directories to be searched for executable files: **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.COM. The file will be searched for and displayed when found.

Type **MAN** to see all available files. They will be displayed in the order of the directories given to MANPATH. If something went wrong in defining MANPATH you will see the help screen again.

Note: Man pages for all commands of SpartaDOS X are available as part of the SpartaDOS X Toolkit. Additionally, builds for several hardware platforms already contain the available man pages. $MANPATH is set to 'CAR:'. If you change MANPATH, keep 'CAR:' in it for the man pages stored in the cartridge.

Type **MAN** to see a list, type **MAN MAN** or even more convenient **MAN MAN | LESS** to read about the usage.

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.

As of SpartaDOS X 4.42 the method of displaying help files has changed. Before, the program contained paging code that worked similarly to MORE, except that it additionally folded long lines.

Now MAN.COM 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 now works just like TYPE, when the /P switch was omitted (in fact, this is exactly what is being done). This is useful, when you want to redirect the 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.

Currently there are three viewers available: the simple TYPE (with the /P switch), MORE, and (a bit better) LESS. LESS is in fact what formerly was the MAN's internal viewer, that has been separated from its program and developed. To register it, 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 fname.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.COM 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.COM 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.

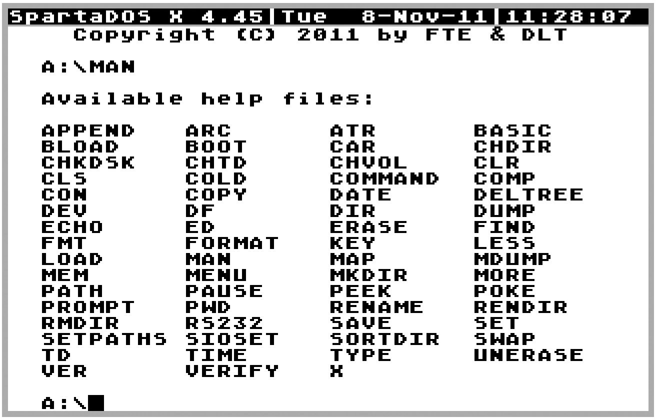
So, in the command line template the filespec's place is marked with the '%' character (when you omit it, the filespec will be simply appended at the end), and spaces are replaced with semicolons. Thus the command to register TYPE /P as an external viewer for MAN.COM looks like this:

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).



```
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
  SETPATHS SIOSET   SORTDIR  SWAP
  TD       TIME     TYPE     UNERASE
  VER      VERIFY   X

  A:\
```

Fig. 28: Available help files for MAN with SpartaDOS X

MAP Command

Purpose
SIO.SYS control.

Syntax
MAP unit [SIO|OS|NORMAL|OFF] [d:]
MAP [/?] | [fname.ext]

Type
External – on device CAR:

Availability
As of SpartaDOS X 4.40.

Remarks
MAP or MAP /? will invoke 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 SpartaDOS communication routines. 'OS' cannot be used, when the DOS is configured to USE OSRAM mode.

MAP 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!

MDUMP Command

Purpose

Display memory in hex and ATASCII.

Syntax

MDUMP $address $len
MDUMP address len

Type

External – on device CAR:

Related

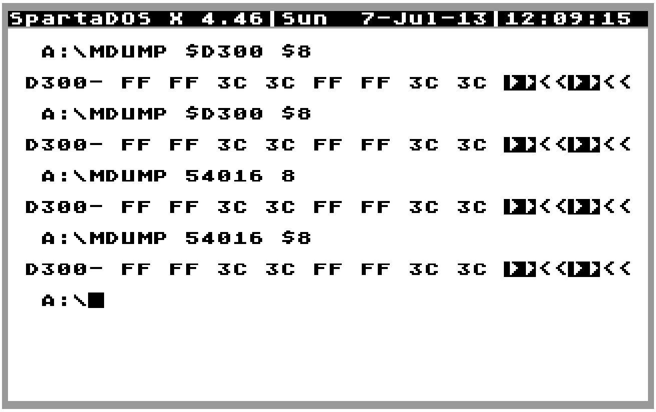
DUMP

Availability

As of SpartaDOS X 4.40.

Remarks

This command does the same to memory, what DUMP does to files. It is useful to check the memory content quickly.



```
SpartaDOS X 4.46|Sun  7-Jul-13|12:09:15
 A:\MDUMP $D300 $8
D300- FF FF 3C 3C FF FF 3C 3C ▶▶<<▶▶<<
 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:\█
```

Fig. 29: Memory dump

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

MEM Command

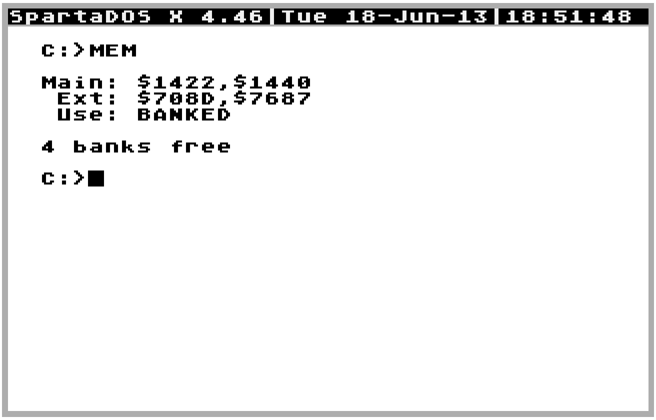
Purpose
This command displays the current memory information.

Syntax
MEM [/X]

Type
External – on device CAR:

Related
CHKDSK, LOAD, MDUMP

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:>█
```

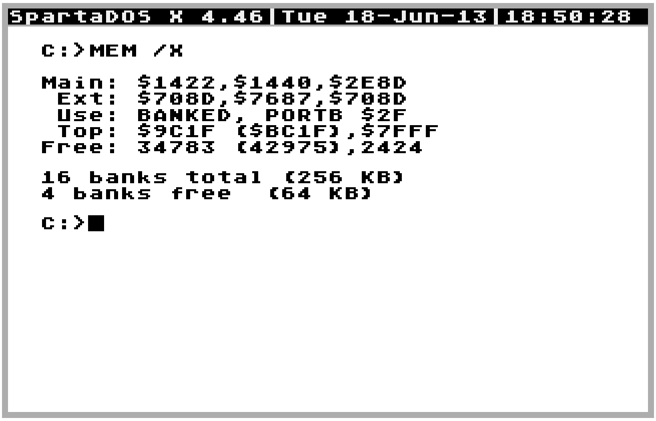
Fig. 30: Memory information

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. The drivers installed in memory are files such as SPARTA.SYS, ATARIDOS.SYS, RAMDISK.SYS, etc.

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.

An example of MEM /X (same configuration as previous example):



```
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. 31: 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 SpartaDOS X. 'Top' shows the highest available address in the main memory (before and after X.COM) and in the ext 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.

Note: If a permanently installed 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, SpartaDOS X may use only one at a time. This is determined at bootup 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).

SpartaDOS X currently supports up to 2 MB of extended memory. The MEM command should work with all known memory extensions and display their status properly. In case of any problems please do not hesitate to contact DLT Team through the web site

http://sdx.atari8.info/

MENU Program

Purpose

This program allows you to select files and then perform COPY, ERASE, RENAME, etc. commands on all selected files. It is similar to other SpartaDOS menu programs, but provides many new features.

Syntax

MENU or just the '∗' key

Type

External - on device CAR:

Related

COPY, ERASE, RENAME, TYPE

Remarks

MENU is useful for operations that include more than one file and is required for single drive copies. It includes three main windows with the commands and prompts displayed below the windows. The "upper left window" is for directories. It will display the subdirectories along with the tree structure showing how they are related. The "upper right window" shows statistics on the logged area. This includes: filespecs, total files, total bytes, tagged files, and tagged bytes. The "lower window" shows the files.

The command menus are broken up into three major classifications: file (File), directory (Dir), and extra (Xtra). The classification is indicated at the lower left of the screen. Toggle between the file and directory command menus by pressing RETURN. You can go to the extra command menu by pressing ESC. To exit from the MENU you press ESC then 'Q'. The '^' character before a menu selection means to hold down the CONTROL key while pressing the selection key.

File command menu includes: COPY, DELETE, EXEC, FILESPEC, LOG, PRINT, RENAME, SHELL CMD, TAG, UNTAG, and VIEW.

Dir command menu includes: AVAILABLE, DELETE DIRECTORY, FILESPEC, LOG, MAKE DIRECTORY, PRINT, SHELL CMD, TAG DIRECTORY, and UNTAG DIRECTORY.

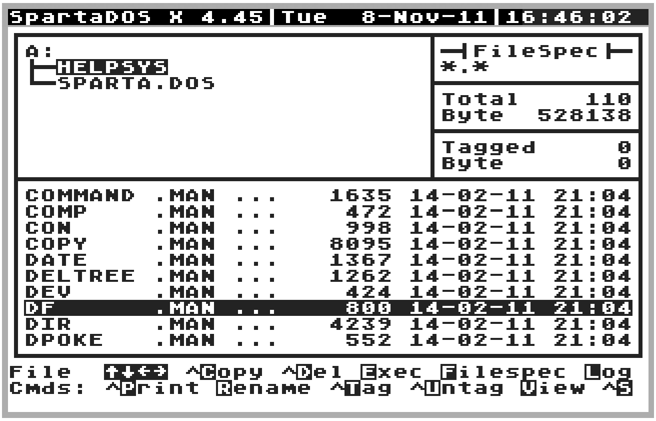
Xtra command menu includes: DISPLAY, QUIT, SORT, and SHELL CMD.

As a special there is the new '^S' (shell command), which allows to execute any command processor command (e.g. DIR) without leaving the MENU program.

The "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).

File Commands
While in the file command menu, use the '↑↓' keys to move the file selector up or down one file at a time or use the '←→' keys to move the file selector up or down one screen at a time. The files shown in the file window are sorted alphabetically by name. They represent the current directory shown in the directory window under the directory selector. (See "Dir Commands".)



```
SpartaDOS X 4.45|Tue   8-Nov-11|16:46:02
A:                              ┤FileSpec├
┣HELPSYS                        *.*
┗SPARTA.DOS
                                Total      110
                                Byte    528138

                                Tagged       0
                                Byte         0
COMMAND  .MAN ...    1635 14-02-11 21:04
COMP     .MAN ...     472 14-02-11 21:04
CON      .MAN ...     998 14-02-11 21:04
COPY     .MAN ...    8095 14-02-11 21:04
DATE     .MAN ...    1367 14-02-11 21:04
DELTREE  .MAN ...    1262 14-02-11 21:04
DEV      .MAN ...     424 14-02-11 21:04
DF       .MAN ...     800 14-02-11 21:04
DIR      .MAN ...    4239 14-02-11 21:04
DPOKE    .MAN ...     552 14-02-11 21:04
File    ↑↓←→ ^Copy ^Del Exec Filespec Log
Cmds: ^Print Rename ^Tag ^Untag View ^S
```

Fig. 32: Menu program screen

C Copy - will copy the file under the file selector. You will be prompted for a destination drive, then a destination path. If copying to the same drive you will also be prompted to insert the destination disk, and then to insert the source disk.

^C **^Copy** - will copy all tagged files. Prompts are like with Copy.

D **Delete** – will delete the file under the file selector. You will be prompted for confirmation.

^D **^Delete** – will delete all files tagged with tag character '◆'. You will be prompted to decide if all tagged files should be deleted with or without confirmation.

E **Exec** - executes the file pointed to by the cursor.

F **Filespec** - allows you to 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** - will allow you to change the logged drive number and/or path.

P **Print** - will print the file currently under the file selector. This is only useful for ASCII text files unless you have a printer driver installed which will print ATASCII graphics characters.

^P **^Print** - will print the files currently tagged. It will send a form feed in between files.

R **Rename** - will allow you to rename the file under the file selector. As a reference, rename prompts you with the present drive number, path, and filename. You can then enter the new filename directly under the old.

^S **Shell Cmd** allows to execute command processor commands.

T **Tag** - will 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** - will tag all files currently logged (in the current directory).

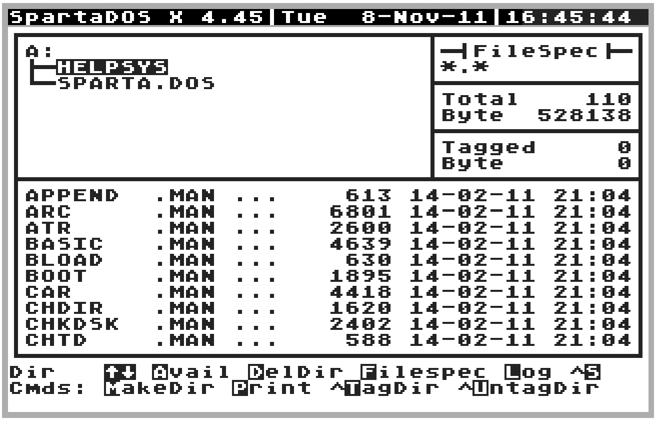
U **Untag** - will untag the file under the file selector. The tag character will disappear and the file selector will move down to the next file.

^U **^Untag** - will untag all files currently logged (in the current directory).

V **View** - will 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".



```
SpartaDOS X 4.45|Tue   8-Nov-11|16:45:44
A:                                  ┤FileSpec├
├HELPSYS                            *.*
└SPARTA.DOS
                                    Total      110
                                    Byte    528138

                                    Tagged       0
                                    Byte         0
APPEND   .MAN ...    613 14-02-11 21:04
ARC      .MAN ...   6801 14-02-11 21:04
ATR      .MAN ...   2600 14-02-11 21:04
BASIC    .MAN ...   4639 14-02-11 21:04
BLOAD    .MAN ...    630 14-02-11 21:04
BOOT     .MAN ...   1895 14-02-11 21:04
CAR      .MAN ...   4418 14-02-11 21:04
CHDIR    .MAN ...   1620 14-02-11 21:04
CHKDSK   .MAN ...   2402 14-02-11 21:04
CHTD     .MAN ...    588 14-02-11 21:04
Dir    ↑↓ Avail DelDir Filespec Log ^S
Cmds: MakeDir Print ^TagDir ^UntagDir
```

Fig. 33: Menu program - directory commands

A **Avail** - will give you the amount of free space available on a drive. You are prompted to enter a drive number and will be shown the free space 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** - allows you to 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. You must go to the "File Cmnds" if you want to do any file operations other than tagging or untagging full directories.

L **Log** - will allow you to change the logged drive number and/or path.

M **Make Dir** - will create a new subdirectory in the current directory selected. After the new directory is created, the system will re-log and you will be back at the root of what was previously logged (always indicated by '>').

P **Print** - will prompt you with two choices: Directory or Tree. Directory will print 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 will print the directory map (tree structure) as displayed in the directory window.

^S **Shell Cmd** allows to execute command processor commands.

T **Tag Dir** - will tag all files in the current directory (under the directory selector).

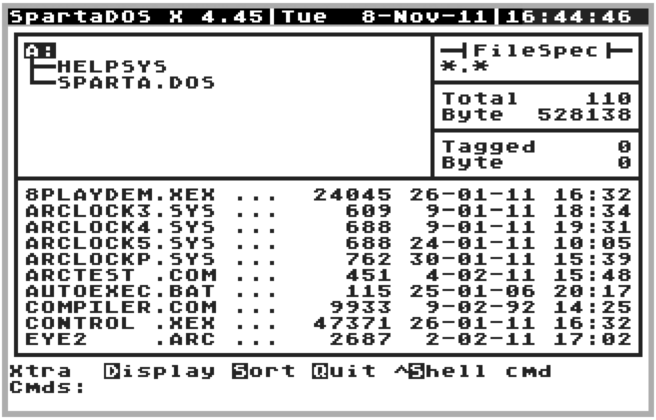
^T **^Tag Dir** - will tag all files in all directories currently logged.

U **Untag Dir** - will untag all files in the current directory (under the directory selector).

^U **^Untag Dir** - will untag all files in all currently logged directories.

Xtra Commands

The "Xtra Cmnds" always takes you back to the previous command menu when finished (except Quit). You can also press ESC to leave this menu.



```
SpartaDOS X 4.45|Tue  8-Nov-11|16:44:46
A:                             FileSpec
├HELPSYS                       *.*
└SPARTA.DOS
                               Total    110
                               Byte  528138

                               Tagged     0
                               Byte       0
8PLAYDEM.XEX ...  24045 26-01-11 16:32
ARCLOCK3.SYS ...    609  9-01-11 18:34
ARCLOCK4.SYS ...    688  9-01-11 19:31
ARCLOCK5.SYS ...    688 24-01-11 10:05
ARCLOCKP.SYS ...    762 30-01-11 15:39
ARCTEST .COM ...    451  4-02-11 15:48
AUTOEXEC.BAT ...    115 25-01-06 20:17
COMPILER.COM ...   9933  9-02-92 14:25
CONTROL .XEX ...  47371 26-01-11 16:32
EYE2    .ARC ...   2687  2-02-11 17:02
Xtra  Display Sort Quit ^Shell cmd
Cmds:
```

Fig. 34: Menu program - Xtra commands

D **Display** - toggles the display in the file window between two types. The default display shows the filename with extension, the status of the three file attributes, the file size in bytes, along with the date and time of creation. This display takes all 38 columns in the file window. The optional display shows two columns of filenames (side by side) with extensions, and their attributes. The attribute letter is displayed if set or a dot if cleared.

Q **Quit** - is the correct way to exit the menu back to the DOS command PROMPT. **Note:** Do not quit or stop operations by pressing RESET. This is a very bad practice that can lead to unrecoverable files.

S **Sort** - will sort the file display by: Name, Ext, Date, Size. This is a forward sort which defaults to name. To permanently sort directories or reverse sort them, use SORTDIR.

^S **Shell Cmd** allows to execute 'command processor' commands.

MKDIR (Make Directory) Command

Purpose
This command creates 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 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. See chapter 8 for more details.

MORE

Purpose
Text Viewer

Syntax
MORE fname[.ext]
MORE <<fname[.ext]

Type
Internal

Related
LESS, TYPE

Availability
As of SpartaDOS X 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, for example:

```
D1:ARC | MORE
```

is an equivalent to:

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

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 from the SpartaDOS X Toolkit.

If MORE is invoked without parameters, it will expect input from CON: and thus act like a memopad. Terminate it with <CTRL><3>; <BRK> or <RESET> abort it.

Chapter 5 contains more information on this topic.

PATH (Set Search Directory) Command

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
You may specify a list of drives and path names separated by semicolons. After this, when you enter a command, SpartaDOS searches the named directories in the sequence you entered them (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 SpartaDOS 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 no path has been specified, the system defaults to:

PATH CAR:

This means search the CAR: device first, then search the current directory. 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.

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.

For more information, see the SET command in chapter 5.

PAUSE Command

Purpose
Suspends system processing and displays the message 'Press RETURN to continue'.

Syntax
PAUSE [n]

Type
Internal

Related
DIR, TYPE

Remarks
You can insert PAUSE commands within a batch file to provide the opportunity to change diskettes between commands or to step through a process, giving you time to read instructions, etc.

To resume execution of the batch file, press the RETURN key.

PAUSE now optionally accepts a number of seconds to wait, ranged from 0 to 65535.

Note: It is very dangerous to change diskettes 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 diskettes, run the batch file from a ramdisk or another drive that will not be changed.

PEEK Command

Purpose
To examine a memory location, perform a HEX conversion or symbol evaluation.

Syntax
PEEK $address
PEEK address
PEEK symbol

Type
Internal

Related
POKE, DPOKE (SpartaDOS X Toolkit)

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 the location in both dec and hex, the dec and hex value of the memory word stored in the location and location+l, and the ATASCII character representing the value of the location.

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

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: Specially if you are not sure what you are doing, it is good practice to always PEEK a location before POKEing a value into it. Usually recovering by POKEing the old value back in is possible (unless the computer has crashed).

POKE Command

Purpose

To change the content of a memory location.

Syntax

```
POKE $location $value
POKE location value
```

Type

Internal

Related

DPOKE (SpartaDOS X Toolkit), PEEK

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.

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

A few examples of POKE 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
```

A good memory map will provide much more information and is a must for programming the ATARI.

Note: Specially if you are not sure what you are doing, it is good practice to always PEEK a location before POKEing a value into it. Usually recovering by POKEing the old value back in is possible (unless the computer has crashed).

PROMPT (Set System Prompt) Command

Purpose
Change the system prompt.

Syntax
PROMPT [prompt string]

Type
Internal

Related
PATH

Remarks
The text in prompt string is taken by SpartaDOS X 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 print current drive letter ('A' through 'O') and a following colon (e.g. 'C:')

N print current drive number ('1' through '9')

P print path on current drive

D print current date

T print current time

R print 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.

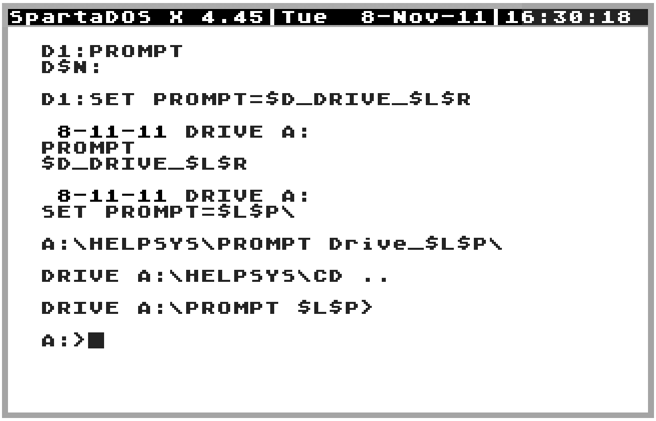
The PROMPT command is just a convenient form of the SET command. **SET PROMPT=$L$P** would perform the same. The default value of the "prompt" variable is 'D$N:', which displays the same prompt like previous versions of SpartaDOS X. If the $PROMPT variable is not defined, SpartaDOS X will prompt with 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 editor lines by moving the cursor and pressing just RETURN.

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 format not recognized by SpartaDOS X. (To use ATARI DOS format disks with SpartaDOS X 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 variable with a **SET 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. 35: Customize your system prompt

PWD Command

Purpose
Outputs a list of current working directories.

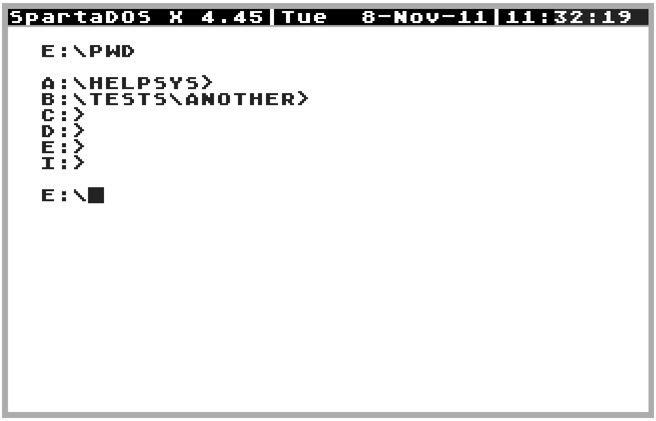
Syntax
PWD

Type
External - on device CAR:

Related
DIR

Availability
As of SpartaDOS X 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. 36: Display the current working directories

RENAME Command

Purpose
This command allows you to 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 wildcarding 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 SpartaDOS X 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.

RENDIR Command

Purpose
This command allows you to change the name of a directory.

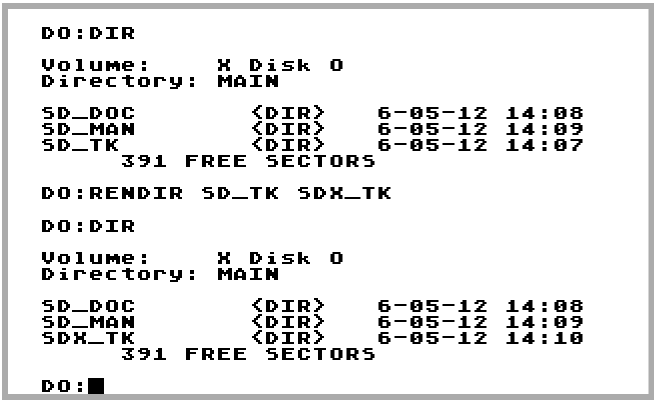
Syntax
RENDIR [d:][path]dir_name_old dir_name_new

Type
Internal

Related
MKDIR, RMDIR

Availability
As of SpartaDOS X 4.40 (as of 4.42 as internal command).

Remarks
The command does to directories, what RENAME does to files. The same rules may 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. 37: RENDIR Command

RMDIR (Remove Directory) Command

Purpose
This command deletes 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.

For related information see the CHDIR, DELTREE, MKDIR, and PATH commands.

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" or "DiskRx" from FTe's SpartaDOS Toolkit can be used to mark such an entry as deleted and not open so that the directory may be removed. The status byte of the directory entry will have bit 7 and bit 3 set. These should be cleared and bit 4 set. Some sectors may be allocated to this file. Those should be deallocated in the bit map. See more in Appendix E.

RS232 (Load RS232 Driver) Command

Purpose

This command loads the RS232 handler from a P:R: Connection or the ATARI 850 interface.

Syntax

RS232

Type

External - on device CAR:

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.

SAVE Command

Purpose

This command saves binary data from memory to disk.

Syntax

SAVE [d:][path]fname[.ext] $address $address
SAVE [d:][path]fname[.ext] address address

Type

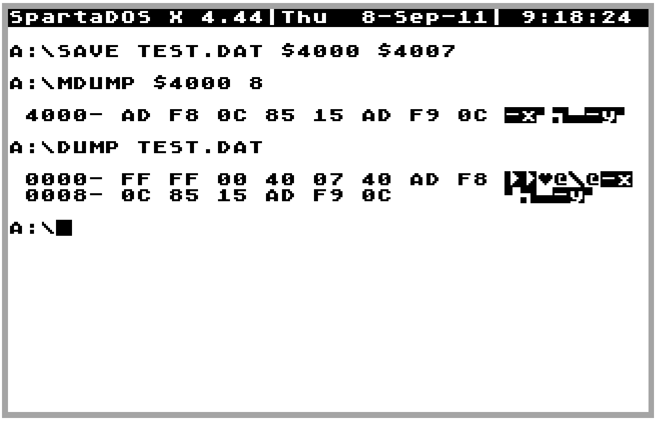
Internal

Related

LOAD, BLOAD

Remarks

Addresses are assumed to be decimal unless preceded by a '$', which indicates hex. This is useful 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 4.44|Thu  8-Sep-11| 9:18:24

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. 38: SAVE Command

SET Command

Purpose

To display the values of all environment variables, and optionally to set an environment variable to a specified value.

Syntax

SET [var[=env_string]]

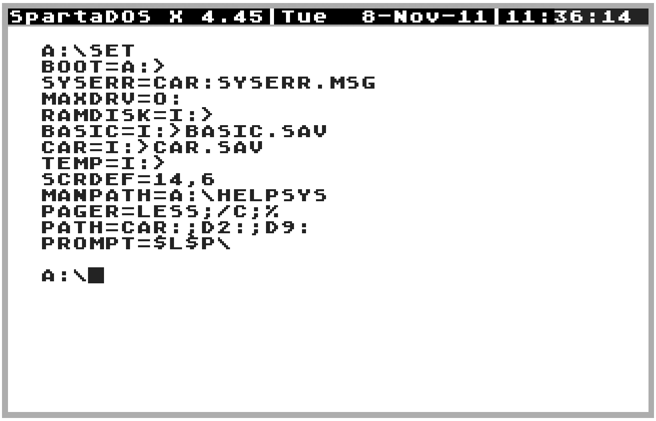
Type

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
MAXDRV=O:
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. 39: 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.

SETPATHS Command

Purpose
To set current directories on the specified drives

Syntax
SETPATHS [d:][path]|fname[.ext]

Type
External - on device CAR:

Related
CHDIR

Availability
As of SpartaDOS X 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.

SIOSET Command

Purpose
SIO.SYS serial speed control.

Syntax
SIOSET [d:] [type [usindex]] or SIOSET NMI [index]

Type
External - on device CAR:

Availability
As of SpartaDOS X 4.40.

Remarks
SIOSET manages an advanced serial protocol control for the SIO.SYS driver. Typically, the serial transmission parameters are determined automatically and 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 them manually.

SIOSET invoked with no arguments displays the current configuration for all drives accessed so far, otherwise a hyphen is noted. The DF command accesses all drives, which helps to see there configuration with SIOSET.

The *type* parameter has the following meaning:

RESET The transmission parameters for the drive are cleared; they will be determined on next I/O request sent to that drive.

NORMAL The drive works at standard baud rate.

XF The drive uses the XF551 protocol.

US The drive uses the UltraSpeed protocol.

INDUS The drive uses the Indus protocol (LDW Super 2000, CA 2001).

When UltraSpeed is used, the additional *usindex* parameter allows to determine the serial speed. For example, the so called 3x SIO mode (3x 19200, i.e. 57,6 kbps) requires $08 as *usindex* 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.

CAUTION: SIOSET does not work, if SIO.SYS was loaded using the '/A' or '/C' option!

Notes: SpartaDOS X high speed SIO driver currently can handle SIO transfers stable as low as ultra speed index 3. The possible SIO speed in your system depends on your hardware setup.

When using the new Ultra Speed driver from the KMK/JZ 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.

SORTDIR Command

Purpose
To sort filenames in directories by name, type, date or size.

Syntax
SORTDIR [d:] [path] [/N|T|S|D|X]

Type
External – on device CAR:

Availability
As of SpartaDOS X 4.41.

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 originates from the SpartaDOS Toolkit and as of SpartaDOS X 4.41 was present on the CAR: device. From version 4.43 on SDX contains a new utility on CAR: with the same name written complete new from scratch to overcome some disadvantages.

SWAP (Swap Drives) Command

Purpose

This command allows you to swap (re-map) your drive configuration.

Syntax

SWAP [d,d] or SWAP [d d]

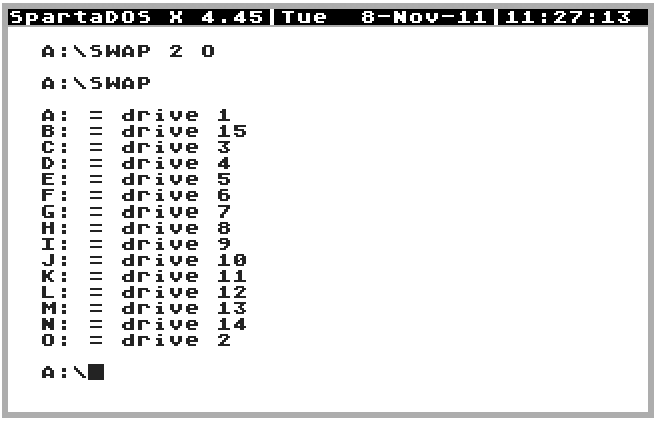
Type

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 4.45|Tue  8-Nov-11|11:27:13

A:\SWAP 2 O

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:\
```

The order is not important, so 2,O 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.

SWAP works in *addition* to the MAP 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.

TD (Time/Date Display) Command

Purpose
This command allows you to turn on and off a time/date display line on top of your screen.

Syntax
TD [ON|OFF]

Type
External - on device CAR:

Related
CHTD, DATE, TIME, DAYTIME

Remarks
This command is much like the KEY command in the way it links into your computer.

You must have either the JIFFY.SYS, ARCLOCK.SYS, RTIME8.SYS, IDEPTIME.SYS or another clock driver installed before you can use this command. It calls one these drivers directly (through the I_GETTD symbol) and will not load without one of these drivers. Some of these drivers are loaded by default unless you are using your own CONFIG.SYS file.

The format in which the time is displayed depends on the value of the $DAYTIME variable (see DATE command).

An additional feature is the 'X' letter in the time/date display line, which reflects the Caps/Inverse state of the keyboard.

See description of Z.SYS (chapter 8) for how to read and set time/date from e.g. BASIC when writing own programs.

Note: If deemed necessary, 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.

TIME Command

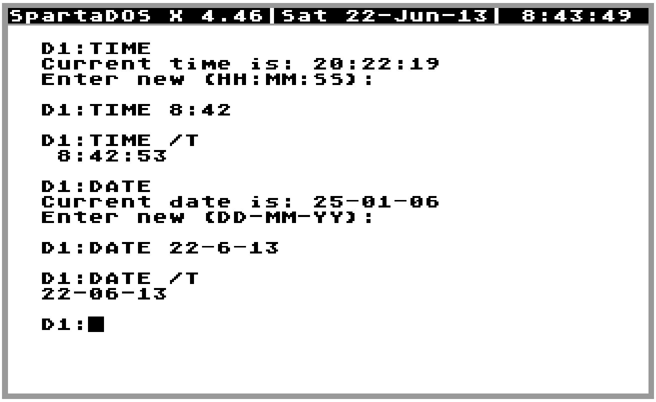
Purpose
This command displays the current time and allows you to set the time.

Syntax
TIME [/T|hh:mm:ss]

Type
External - on device CAR:

Related
CHTD, DATE, TD

Remarks



```
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. 40: Using the TIME command

Enter the new time or just 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 used with /T parameter only the current time 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.

Without a proper clock driver installed this command will produce meaningless results. SpartaDOS X currently provides several drivers for real time clocks and for the ATARI software clock (→ JIFFY.SYS). By default, one of these drivers will be installed during boot up, but can be overridden by creating a custom 'CONFIG.SYS' file to change the preset. See chapter 8 (→time keeping drivers) for more details.

Notes: SpartaDOS X V 4.4x uses a 24 hour format, whereas older versions of SpartaDOS use a 12 hour format with AM/PM.

Sometimes a realtime clock 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 from the SpartaDOS X toolkit.

TYPE Command

Purpose
This command displays the contents of a specified file.

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

Type
Internal

Related
COPY, DUMP, MENU, PAUSE, LESS, MORE

Remarks
You may display any file and are not limited to a maximum line length (as was the case with SpartaDOS 3.2). Press <CTRL-1> to stop and start the display. You may specify attributes as in the DIR command - the default attributes are '-HS'. (See the DIR command for a description of the attributes.)

If you include a '/P' parameter, the TYPE command will wait for a key press after each 23 lines of text.

UNERASE Command

Purpose

This command restores files previously erased (if possible).

Syntax

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

Type

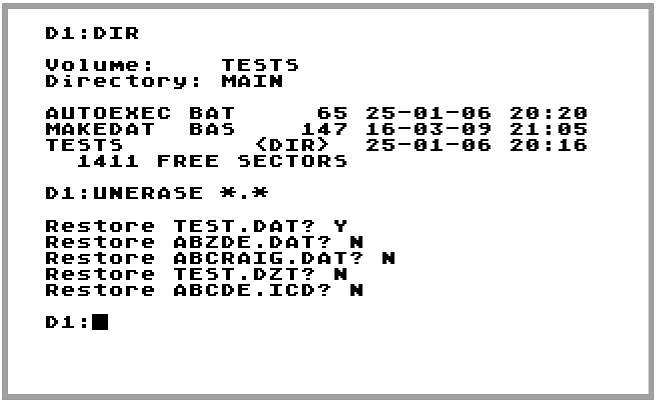
External - on device CAR:

Related

ERASE

Remarks

Wildcards are permitted. You will be asked for each recoverable file matching the filespec you entered, if you wish it restored or not. If you think that erased files exist in your directory that are not seen 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.DZT? N
Restore ABCDE.ICD? N

D1:█
```

Fig. 41: Restore files using UNERASE

Note: UNERASE.COM in SpartaDOS X 4.2x contains a serious bug, that is fixed now. The test distributed with Nelson Nieves' NNTOOLS now passes without errors.

VER command

Purpose

To display the current version number and date of the cartridge.

Syntax

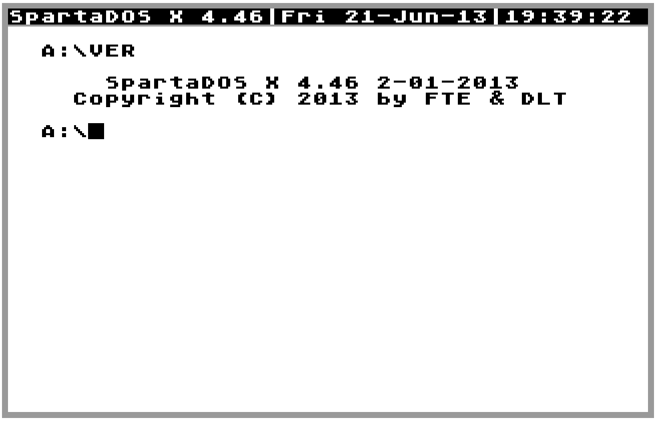
VER

Type

Internal

Remarks

This command will show the version number, revision date, and copyright notice as displayed when the cartridge is booted.



```
SpartaDOS X 4.46|Fri 21-Jun-13|19:39:22
 A:\VER

    SpartaDOS X 4.46 2-01-2013
  Copyright (C) 2013 by FTE & DLT

 A:\█
```

Fig. 42: Display the SpartaDOS X version information

VERIFY Command

Purpose
To turn write verify on or off.

Syntax
VERIFY ON|OFF

Type
Internal

Remarks
When ON, SpartaDOS performs a verify operation following each disk write operation, to verify that the 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.

X Command

Purpose

Execute a program which requires that no cartridges are installed (such as "DiskRx", EXPRESS, most binary files, etc.)

Syntax

X [/C] [d:][path]fname[.ext] [parameters]

Type

External - on device CAR:

Remarks

There are four possible environments for a command to be run:

1 with internal BASIC present (via BASIC command)

2 with a CARtridge present (via CAR command)

3 with the SpartaDOS X library enabled (just type the program or command name)

4 with no cartridges present (via the X command)

The first three modes use the SpartaDOS X 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 commands 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 - SpartaDOS 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, available as of SpartaDOS X 4.42, causes the 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.

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 SpartaDOS X cartridge and the external cartridge, if present.

5 The Command Processor – Advanced Features

The SpartaDOS X 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 SpartaDOS or have been greatly enhanced over prior versions of SpartaDOS.

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 **#DISKRX** will have the same effect as **X DISKRX**.

Batch Files
Batch files are simply a list of SpartaDOS 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.

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. As of SpartaDOS X 4.41 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 that accepts two files on input and creates an output file containing the content of both source files (we will call this file 'TEST.BAT').

```
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 above 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.

Default Batch File
When the command processor is first entered, it tries to run a batch file called 'AUTOEXEC.BAT'. You should put any system setup commands that you may need in this file.

The variable $BATCH 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 $BATCH can be used to cause a batch file with a different name to be executed. Use the line:

SET BATCH=d:filename.BAT

in CONFIG.SYS.

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.

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. Example usage:

```
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
```

For example, this can be saved as a batch file named X.BAT somewhere in your $PATH.

Then, typing at the command prompt **-X FILES** allows to unpack the archive FILES.ARC into a subdirectory automatically created for this purpose.

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.

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.

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".

Note: As of SpartaDOS X 4.42 the syntax of the comparisons has been changed to mimic MSDOS.

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
```

This is the definition. To give the label’s name as a parameter to the 'GOTO' keyword you have to 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.

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 SpartaDOS X 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).

As of SpartaDOS X 4.42 an extension 'ENV.SYS' is available, which provides as much as 15 KB 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.

Note: The 'INKEY' command should not be confused with the 'INKEY' operator, which is part of the 'IF' command.

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.

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.

I/O Redirection

On the command line you may redirect the standard input and output of SpartaDOS X. With SpartaDOS 3.2, redirection was done using batch files (for input redirection) and the PRINT command (for output redirection). SpartaDOS X implements I/O redirection in a totally different manner. Batch files are no longer considered as "input redirection" – they are only read by the command processor and are not system wide (i.e. you can not feed input to BASIC through a batch file). The PRINT command has been eliminated.

With SpartaDOS X, you may divert 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. For example, the command 'DIR >>PRN:' redirects the output from the DIR command to the printer (the directory listing will not appear on the screen). An alternate way to copy a file would be 'TYPE fname >>dest'.

This will be slower than the copy command and will not copy the date to the new file.

'BASIC <<AUTOGO' will run the BASIC program 'START.BAS' if the text file 'AUTOGO' contains the line

```
RUN "D:START.BAS"
```

As an example of output redirection, 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
```

Example: If you name the above batch file MORE.BAT and want to read the directions printed by the ARC program, use **-MORE CAR:ARC.COM**.

Pipes

SpartaDOS X as of version 4.42 features the mechanism of pipes, known from MSDOS 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 SpartaDOS X 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.

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, if the external INVERSE command displays the given text in inverse video, you can do this like **ARC | INVERSE | TYPE - /P**. The INVERSE command has been moved to the SpartaDOS X Toolkit.

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.

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 you to keep often used utilities out of the way in subdirectories. This is particularly useful if you own a hard drive, since the main directory of D1: can get very cluttered.

You can examine the $PATH variable by typing **PATH** with no parameters at the CP prompt. The default $PATH is 'CAR:'.

The search path may be changed by "PATH path1;path2;...;pathn" at the CP prompt. 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 and then the current directory of the default drive.

If a path is specified on the command line, this search path will *not* be used.

The search path can also be used for batch files, the X command, the BASIC command and the CAR command.

It is also possible to access the 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.

Because of these changes, it is a good idea to make the current directory the second entry in the path. Changing the above example to do this would produce 'PATH CAR:;;A:\DOS\;A:\TOOLKIT\;D9:>;A:>;:>' where the two semicolons after 'CAR:' signify that current directory should be searched.

Automatic evaluation of environment variables

As of SpartaDOS X 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

SpartaDOS X Functions from BASIC

Many SpartaDOS 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; 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 operating system for the screen editor, so it should not be used.

An 'ATARI DOS disk' is one initialized in ATARI DOS 2 format, whether in single, enhanced (dual), or double density, as produced by ATARI DOS 2.0S and 2.5, MYDOS, other DOS 2 clones, and the SpartaDOS X Formatter when used in ATARI DOS mode.

'd:', 'path', and 'fname.ext' refer to any legal SpartaDOS X device identifier, pathname, and filename with extension as defined in chapter 4.

Notes on the Default Drive

Please remember that D: *from BASIC or another language refers to the* default drive, *not necessarily drive #1.* From the command processor, D: refers to drive #4. With most other DOS types including earlier versions of SpartaDOS, *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 SpartaDOS X.

Accessing the„Kernel“Through CIO

The D: device available through the CIO with SpartaDOS X is not just the disk drive handler; it is the handler for the SpartaDOS "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.

Open File

Purpose
To open a disk file for access through SpartaDOS X.

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

Remarks
This command opens a disk file through SpartaDOS X. 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.

As of SpartaDOS X 4.42 these additional attributes are available:

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 SpartaDOS 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
```

Directory formatting attributes

As of SpartaDOS X 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)
	0	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).

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 SpartaDOS X 4.41 the short directory format (compatible to ATARI DOS 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.

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 SpartaDOS 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 ATARI DOS disks if the ATARIDOS.SYS driver is installed. The driver translates the ATARI directory format into SpartaDOS 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.

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.

Rename File(s) (RENAME)

Purpose
To change the name of a file or 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 SpartaDOS X 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: SpartaDOS X has an extremely powerful RENAME function. As of version 4.4x new routines have been implemented to make sure, that you cannot give the same name to more than one file in the same directory.

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.

Erase File(s) (ERASE)

Purpose
To 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.

Protect File(s) (ATR +P)

Purpose
To prevent a file or 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.

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.

Set File Position – POINT

Purpose

Allows direct access to specific points within a file (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 ATARI DOS and compatibles, that use an absolute physical disk position (sector and offset into sector) for the NOTE and POINT functions, SpartaDOS X 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 SpartaDOS, you should notice that NOTE and POINT now work the same way with ATARI DOS disks as they do with SpartaDOS disks. POINT will *not* use sector number and offset regardless of disk format.

Using NOTE and POINT with SpartaDOS X on an ATARI DOS 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 ATARI DOS media than from SpartaDOS disks. NOTE and POINT tables created by other DOS types (including earlier versions of SpartaDOS) for files on ATARI DOS disks will no longer be valid.

Sparse Files

On a SpartaDOS 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 SpartaDOS X routine, corresponding to this function, has been rewritten, so that random access to very long files (greater than 500k) should now work up to four times faster than before.

Get Current File Position – NOTE

Purpose

To 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 SpartaDOS 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 SpartaDOS Construction Set manual will still work but is redundant.

Get File Length

Purpose

To 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 the file.

Load a Binary File (LOAD)

Purpose
To 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 execute it using the INIT and/or RUN vectors. The IOCB should be closed. Loading a binary file from an ATARI DOS disk will take much longer than loading the same file from a SpartaDOS format disk. As of SpartaDOS X 4.41, if X is 0, the file will be loaded to the memory and executed. If X is 128, the file will be loaded without execution.

Note: Versions before SpartaDOS X 4.4x 'forgot' to behave like this even it has been already implemented with SpartaDOS 3.2. They did not execute the loaded file, and there was no separate 'load/execute' function either. So we restored the behavior known from SpartaDOS 3.2.

Create a Directory (MKDIR)

Purpose
To 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 SpartaDOS formatted disks.

Delete a Directory (RMDIR)

Purpose
To 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.

Change Current Directory (CHDIR)

Purpose
To 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.

Set Boot File (BOOT)

Purpose
To establish the file that will be loaded when the computer is initialized without the use of SpartaDOS X.

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 SpartaDOS X cartridge is not used. With earlier versions of SpartaDOS, the primary use of this was to cause the *.DOS file to be booted. With SpartaDOS X, the uses of this command are limited. The IOCB should be closed and a SpartaDOS format 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.

Set Attributes (ATR)

Purpose
To 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.

Format a Disk (FORMAT)

Purpose
To 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 SpartaDOS X disk formatter menu. From this menu, disk number, format, size, skew, etc. may be selected. Using the ESC key, the program will exit 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! Formatting a medium may destroy all existing data on a floppy disk. Hiding or protecting a file will not save it from being destroyed during a format.

For more 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.

Get Disk Information (CHKDSK)

Purpose
To read information about a disk

The next 2 paragraphs contain information from SpartaDOS X 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 ATARI DOS 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 ATARI DOS format disks (8 bytes)
+14	=	volume sequence number, 0 if ATARI DOS format
+15	=	volume random number, 0 if ATARI DOS format
+16	=	unused byte

Changes to this function implemented with SpartaDOS X 4.4x:

The earlier versions of SpartaDOS support disks with 128- and 256-byte sectors – SpartaDOS X v. 4.4x adds support for 512-byte sectors and theoretically even larger.

The CHKDSK function returns 17 bytes described in the above paragraphs. 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, Dual) and 0 for 256 bytes per sector (Double). Originally it is, of course, just the low byte of the actual sector size value (128=$0080, 256=$0100) – and encoding sector sizes larger than 256 bytes is impossible this way.

Because of that, the interpretation of this byte in the SpartaDOS X 4.4x has changed, 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.

Just as before, the number 128 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, like below (*=not supported yet):

Size (BPS)	Hex Value	Encoded As	Remarks
128	$0080	$80(128)	1)
256	$0100	$00(0)	1)
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)	

1) backward compatibility to SpartaDOS X 4.1x and 4.2x.

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
```

Get Current Directory Path (CHDIR)

Purpose

To 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 .equ $E456
iccom .equ $0342
icbal .equ $0344
icbah .equ $0345
icbll .equ $0348
icblh .equ $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
```

SpartaDOS User Accessible Data Table

Several SpartaDOS variables have been made 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 SpartaDOS except where noted. Locations COMTAB, ZCRNAME, BUFOFF, COMFNAM, and LBUF are also supported by OS/A+ and DOS XL.

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
SpartaDOS X 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
This is a pointer to the SpartaDOS 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
SpartaDOS X only. A three byte number here will be converted by the "misc_convdc" routine to a string at DECOUT (see Page Seven "Kernel" Values).
A four byte number here will be converted by the misc_conv32 routine to a string at DECOUT2. See 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 iden-

tifier 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 SpartaDOS X.

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 on SpartaDOS 3.x. SpartaDOS X disabled and replaced them with the new triad of DATE/TIME/DATESET. This was the reason, why SpartaDOS 3.2 copy programs under SpartaDOS X 4.2x could not control timestamps in the files copied.

SpartaDOS X 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 SpartaDOS X 4.4, 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
SpartaDOS X only. $FF if the computer is an ATARI 800. Zero otherwise.

NBANKS COMTAB+29
SpartaDOS X only. The number of free expansion memory banks. This is the number also shown by the MEM command.

BANKFLG COMTAB+30
SpartaDOS X only. $FF if USEing BANKED. Zero otherwise.

OSRMFLG COMTAB+31
SpartaDOS X 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 SpartaDOS X "kernel".

An Example
The following assembly language program demonstrates one way to read the SpartaDOS 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.

```
;CIO and IOCB equates
ciov .equ $E456
iccom .equ $0342
icbal .equ $0344
icbah .equ $0345
icbll .equ $0348
icblh .equ $0349
write .equ $09
; SpartaDOS equates
comtab .equ 10
zcrname .equ 3
bufoff .equ 10
comfnam .equ 33
```

```
; The program.
*=$4000                 ; or wherever.
Init                    ; patches our crunch routine to
        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 CIO 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      ;run vector
```

Decoding the drive identifier

SpartaDOS X 4.4x supports an increased number of fifteen drive identifiers. D1: to D9: or DA: to DI: as known from SpartaDOS 4.2x. The six additional drives (above D9: or DI:) are identified by drive letters only from DJ: to DO:.

The convenience known from the previous SpartaDOS 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 "new" 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 SpartaDOS X only, the routine may be greatly simplified:

```
dsk2asc
        lda dunit
        ora #$40
        rts
```

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 SpartaDOS 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 SpartaDOS 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.

Vectors Under the OS ROM

The vectors are created under the OS ROM to secure some backward compatibility with SpartaDOS 3.2 and earlier versions. There is no need to use them in SpartaDOS X, 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.

Here is a list 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†	VINITZ	_INITZ	kernel
$FFCF†	VINITZ2	-/-	-/-
$FFD2	VXCOMLI	XCOMLI	kernel
$FFD5†	VCOMND	-/-	-/-
$FFD8†	VPRINT	PRINTF	kernel
$FFDB†	VKEYON	I_KEYON	KEY.COM

The vectors are not used by SpartaDOS X 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 SpartaDOS X they may disappear completely (the cross in the table marks the locations already obsolete in SpartaDOS X 4.4x).

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 SpartaDOS X 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.

Page Seven "Kernel" Values

Several page seven locations allow for "kernel" operation to be accessed. While it is beyond the scope of this manual 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)
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 lsio 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 SpartaDOS X 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 SpartaDOS, 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.

Using the CON: Drivers in Own Programs

Note: The following information is valid for the drivers bundled with the SpartaDOS X 4.42 or newer versions.

The screen editor provided by the CON64.SYS or CON80.SYS (SDX Toolkit) 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.

Detecting the extension

The following call:

```
XIO 33,#0,12,0,"E:"
```

should return a 1 (success), if either CON64.SYS or CON80.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 CON80.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.

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.

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 loose 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.

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.

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

SpartaDOS File System (SDFS)

The SDFS version 2 is the standard format used by all versions of SpartaDOS 2.x, 3.x SpartaDOS X, BeweDOS, Micro-SpartaDOS and RealDOS. With SDX 4.4x an advanced version 2.1 has been introduced. SDFS version 1.1 from SpartaDOS 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.

Boot Sectors

As with most other DOS types for the 8-bit ATARI computer, 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 SDFS-compatible DOS if specified (with the BOOT command). This code is not used with SpartaDOS X. 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.

SpartaDOS 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 SpartaDOS. 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 SpartaDOS X 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 SpartaDOS 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 old XL Operating System 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 KB 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 ATARI 8-bit 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 SpartaDOS 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 RAMdisk 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. SpartaDOS 1.1 has a $11, SpartaDOS 2.x, 3.x, SpartaDOS X 4.1x and 4.2x all have a $20 here, SpartaDOS X 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.

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.

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. See the "Programming With SpartaDOS X" chapter at the POINT command for a description of sparse files.

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).

When a directory is opened in unformatted or raw mode (see Programming with SpartaDOS X) 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 SpartaDOS. 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).

Exploring Disks

The best way to become familiar with the SDFS is to use a sector editor and a test medium to explore. "DiskRx", the SpartaDOS disk editor from FTe's SpartaDOS Toolkit, is an excellent sector editor, tailored specifically for SDFS media. It will identify boot, bit map, sector map, directory and data sectors, but cannot handle 512 bytes per sector partitions.

Alternatively, 'Eddy' from the SpartaDOS X Toolkit is a very good tool and capable of handling sectors of 512 bytes in size.

A good combination of understanding the SDFS media structure and the editor used, can prove to be invaluable for recovering files from disks with bad sectors or damaged directories.

PERCOM Extensions

SpartaDOS X 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 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).

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
dunit   = $0301
dcmnd   = $0302
dstats  = $0303
dbufa   = $0304
dtimlo  = $0306
dbyt    = $0308
daux1   = $030a
daux2   = $030b
jsioint = $e459
buffer  = $0400        ;cassette buffer

getbootsize
            sta dunit
            lda #$31
            sta ddevic
            lda #'N     ;READ PERCOM
            sta dcmnd
            lda #$40
            sta dstats
            lda #<buffer
            sta dbufa
            lda #>buffer
            sta dbufa+1
            lda #$07
            sta dtimlo
;amount of data: 12 bytes
            lda #$0c
            sta dbyt
            lda #$00
            sta dbyt+1
;this should be zeroed because of
;some floppy turbo systems (e.g.
;Top Drive, TOMS Turbo)
            sta daux1
            sta daux2
            jsr jsioint
            bpl success
;error 139 means that the drive
;does not know READ PERCOM cmd
;so it can only do 128 BPS
;(it is an ATARI 810 or 1050)
            cpy #139
            beq a810
;any other error is just an error
            cpy #$00
            rts
;unmodified 810 or 1050,
;128 bytes per sector
;a810
            lda #$80
            ldx #$00
            ldy #$01
            rts
;success
;low byte of the sector size
            lda buffer+7
;high byte of the sector size
            ldx buffer+6
;if the BPS < 512, return 128
            cpx #$02
            bcc a810
;or the returned value otherwise
            ldy #$01
            rts
```

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 enough memory left to program with or load a particular application. Therefore it is recommended to have at least 128 KB of RAM in your XL/XE machine and reserve some extended memory for DOS (USE BANKED in the CONFIG.SYS file).

The Boot Drive

As of SpartaDOS X 4.40 the boot drive number is forced to D1: only when it was not yet determined upon entering the DOS initialization routines. Normally, the XL Operating System 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:.

When SpartaDOS X boots, it has certain defaults - in fact, it contains a text file of configuration information. You may write your own file and override the default configuration "file". The file you create is called "CONFIG.SYS" and should reside as a text file on the boot drive when you boot. It must be on a SpartaDOS format medium in the "MAIN" directory.

CONFIG.SYS File

The "CONFIG.SYS" file basically consists of commands. The current four commands are

```
USE  OSRAM|BANKED|NONE
SET  var=env_string
DEVICE  driver
MERGE  [d:][path]fname[.ext]
```

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 RAM from $4000-$7FFF in memory expanded ATARI 8-bit computers. NONE refers to low memory, above the normal low memory part of SpartaDOS X and below program memory. USE NONE will probably be incompatible with many programs, since it uses up a great deal of main memory that could be used by applications. Any line beginning with a semicolon (;) is considered a comment and ignored.

The default CONFIG.SYS file is

```
DEVICE SPARTA OSRAM
DEVICE SIO
DEVICE ATARIDOS
DEVICE INDUS 4
DEVICE RTIME8 (or IDEPTIME for KMK/JZ IDE V 2.0 Plus)
DEVICE JIFFY
DEVICE RAMDISK
```

The default RAM usage is as follows:

OSRAM	Stock XL/XE computer
BANKED	Banked memory expanded computer
NONE	400/800 (48 KB) with no expanded memory

The default CONFIG.SYS file is to be found on the CAR: device. It will be read from there, when no user-defined CONFIG.SYS is found on the boot disk. Type TYPE CAR:CONFIG.SYS to see the default configuration file in your current version of SpartaDOS X.

Notes: The size of the OSRAM memory area is 7 KB ($E400-$FFBF) and the BANKED memory area is 16 KB ($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 KB of RAM). If you do choose OSRAM, you may use the 4 KB of RAM from $C000-$CFFF as buffers for the SPARTA.SYS driver (see SPARTA.SYS 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).

As of SpartaDOS X 4.42, 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 your system setup requires a tailored default CONFIG.SYS in CAR: device, please see SDX Imager on http://sdx.atari8.info/.

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.

DEVICE Command

The DEVICE command will load installable drivers such as SPARTA.SYS, RTIME8.SYS, etc. Each of these drivers is documented in this section. Note that order is important (e.g. SPARTA.SYS must load before ATARIDOS.SYS).

If you hold down the OPTION key 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.

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 SpartaDOS X device drivers. Its use is optional, but *when it is 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 KB in size. Holding <OPTION> during system startup disables merging.

Basic usage
Routines from the operating system 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' will be recognized automatically. Additional use of the config selector enhances the possibilities of the basic usage. See respective section for more details.

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 SDX driver 'SIO.SYS', and then load a file named DEFAULTS.CFG from the subdirectory 'CONFIGS' on drive 'D3:'. That file should contain actual 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 SpartaDOS X still cannot boot from non-OS-compliant drives.

Special usage

When using non-OS-compliant 'boot' devices SpartaDOS X 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 SPARTA.DOS directory 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 please see information about the SDX Imager on http://sdx.atari8.info/. If the SpartaDOS X 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.

Config Selector

SpartaDOS X 4.4x 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 regular SpartaDOS disks/partitions.

At boot time the main directory of the boot medium is being searched for a subdirectory named SPARTA.DOS. When it is found and contains *.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. Alternatively, arrow keys or number keys (1 to 9) are legal to select the desired CFG file. The choice needs to be confirmed by pressing RETURN, otherwise the process is paused. Pressing the ESC key or waiting a few seconds without pressing any key causes the DOS to close the menu. The standard procedures using the default CONFIG.SYS file will be executed then by SpartaDOS X.

Notes: If the directory SPARTA.DOS is detected, a CONFIG.SYS file in the main directory of the boot drive is ignored.

As an additional feature the code is re-arranged so that the last selection will be stored in NVRAM, which is currently supported by KMK/JZ IDE 2.0 Plus and Ultimate 1MB. The respective SpartaDOS X builds contain this feature, which remembers the user's last selection and presents 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.

Character Sets

When SpartaDOS X 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.

Important system variables

Some global system settings are controlled with environment variables (type SET and RETURN 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 SpartaDOS X 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. Returning to BASIC will cause the memory to 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

SpartaDOS 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 SpartaDOS X 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). Currently the default format equals EU.

$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 and background respectively. The command processor will set these colors once loaded only when the variable is defined. Setting the variable itself by 'SET SCRDEF=fc,bc' does nothing. For it to take effect, one has to reload the COMMAND.COM e.g. with EXIT. **Note:** EXIT aborts batch processing.

$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.

DRIVERS

With SpartaDOS X 4.4x a lot of all new features have been introduced to the ATARI 8-bit 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 DOS system.

FILE SYSTEM DRIVERS

SPARTA.SYS Driver

Purpose

This is the SpartaDOS format diskette 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 device CAR:

Remarks

This is the largest of all the drivers and contains 3 subprograms, these are

- the SpartaDOS "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 (as in 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.

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 SpartaDOS X 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.

ATARIDOS.SYS Driver

Purpose
This driver contains the code to recognize ATARI DOS 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 device CAR:

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 ATARI DOS 2 including subdirectories of MYDOS up to a size of 16 MB (65535 sectors with 256 bytes). It supports the extended sectors of DOS 2.5 for read only.

It does not support the ability to create (MKDIR) a directory, delete (RMDIR) a directory, or set working directory (CD) on MYDOS media. ATARIDOS.SYS does not provide 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.

BLOCK I/O DRIVERS

SIO.SYS Driver

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 device CAR:

Remarks
One of the SIO drivers must be included in the 'CONFIG.SYS' file. SIO.SYS contains all the code to handle high speed SIO operations with the US Doubler, Indus GT, Happy, Speedy, XF551 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/JZ 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 the SIOSET command (see chapter 4).

Additionally, there is now a built-in mechanism for the "mapping" of drives, accessible by the MAP command (see chapter 4).

If all that is waived to go for the maximum free memory, SIO.SYS can be loaded with the '/C' option (as "Classic"). This will load the old-fashioned SIO.SYS driver as known from SpartaDOS X 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 only occupy a minimum of memory, but the transmission parameters will depend on the OS capabilities.

SIO /A works in any memory configuration. The limitation in ATARI SIO usage, when the memory was configured to USE OSRAM with buffers located under the ROM has now been lifted.

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.

INDUS.SYS Driver

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 device CAR:

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 on higher drives, edit the CAR:CONFIG.SYS and change or remove the parameter digit in DEVICE INDUS line.

RAMDISK.SYS Driver

Purpose

This is the SpartaDOS X ramdisk driver. Drive number and size of a maximum of three ramdisks may be selected.

Syntax

DEVICE RAMDISK [drive][,nbanks] [/S]

Type

External - on device CAR:

Remarks

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 KB RAM banks in the used 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 is

- XL/XE: all available windowed RAM minus 4 banks (64 KB) reserved for BASIC XE, and minus 1 bank used by SpartaDOS X for DOS routines and drivers.

- 800: all available windowed RAM minus 1 bank used by Sparta DOS X 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 65816 mods.

At the beginning of chapter 8 the default system configuration is shown as if you do *not* have a custom CONFIG.SYS file on your boot drive. If you have more than 64 KB of extended memory in your XL/XE computer, SpartaDOS X will 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. One can write a custom CONFIG.SYS specifying USE OSRAM to 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 SpartaDOS X uses* (when there are more than 64 KB of extended memory) and assigning the ramdisk to drive O:.

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.

If you experience problems with your extended memory configuration as a ramdisk, please e-mail the DLT team for help.

Notes: If you hold down the OPTION key when booting the computer, any custom CONFIG.SYS 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.

RAMDISK.SYS has no effect on Multi I/O (MIO) extended memory.

PBI.SYS Driver

Purpose
Provide additional support for PBI devices.

Syntax
DEVICE PBI

Type
External - on device CAR:

Availability
As of SpartaDOS X 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 don't handle PBI bus devices with a driver installed.

TIMEKEEPING DRIVERS

Without any clock driver installed, the TIME/DATE commands respond with giving the time and date of the last file loaded. In most cases this will be the SpartaDOS X revision time and date, which do not advance.

ARCLOCK.SYS Driver

Purpose
Driver for the ARC clock.

Syntax
DEVICE ARCLOCK [/F]

Type
External - on device CAR:

Availability
As of SpartaDOS X 4.40.

Remarks
This is the driver for the battery-backed real time clock called ARC (ATARI Real Clock). ARCLOCK.SYS now performs the clock recognition and does not load when none is found.

The option '/F' (force) bypasses this test.

Notes: If the driver does not recognize your working ARC and responds "ARC clock not present", please try the /F switch.

ARCLOCK.SYS resets the clock to "1-Jan-2000 00:00:00" when it discovers that the current setting is invalid.

IDEPTIME.SYS Driver

Purpose
Driver for the realtime clock residing on the KMK/JZ IDE V 2.0 Plus interface.

Syntax
DEVICE IDEPTIME

Type
External - on device CAR:

Availability
As of SpartaDOS X 4.44.

Remarks
If no KMK/JZ IDE 2.0 Plus is plugged into the PBI/ECI port, this handler will not load.

RTIME8.SYS Driver

Purpose
This is the R-Time 8 driver for SpartaDOS X.

Syntax
DEVICE RTIME8

Type
External - on device CAR:

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.

Note: RTIME8.SYS does not load when Maxflash8 hardware is used. This has been introduced to prevent hang-up caused by hardware conflicts.

Z.SYS Driver

Purpose
Provide a Z: device compatible with SpartaDOS 3.2.

Syntax
DEVICE Z [/I|S]

Type
External - on device CAR:

Availability
As of SpartaDOS X 4.40.

Remarks
A 'Z:' device is created in the OS handler table to offer a simple interface to SpartaDOS timekeeping functions, accessible e.g. from a BASIC program. The device is compatible with the driver for SpartaDOS 3.2 (ZHAND.COM). Therefore software using it should have no problems accessing the desired functions under SpartaDOS X 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.

JIFFY.SYS Driver

Purpose
This is the jiffy clock driver for SpartaDOS X.

Syntax
DEVICE JIFFY

Type
External - on device CAR:

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.

SCREEN DRIVERS

XEP80.SYS Driver

Purpose
80 column display using the ATARI XEP80 adaptor.

Syntax
DEVICE XEP80 [1|2] [/P|/N]

Type
External - on device CAR:

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.

Before SpartaDOS 4.4x the XEP80.SYS driver contained a bug, that prevented it from working on PAL computers. The current version recognizes such machines correctly.

Additionally, you can force either display mode by adding switches to the XEP80.SYS command line: [/P] for PAL or [/N] for NTSC.

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 on-line 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 good monochrome monitor is highly recommended for use with the XEP80.

QUICKED.SYS Driver

Purpose
Provides an accelerated screen output in Graphics 0.

Syntax
DEVICE QUICKED

Type
External - on device CAR:

Availability
As of SpartaDOS X 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 CON80.SYS (SpartaDOS X Toolkit), please be aware of the right order in CONFIG.SYS, where DEVICE QUICKED must precede DEVICE CON64 and/or DEVICE CON80 to get both working properly. And it is recommended to call DEVICE RAMDISK thereafter, otherwise one bank of extended memory is wasted.

CON64.SYS Driver

Purpose
This provides a 64 column display.

Syntax
DEVICE CON64

Type
External - on device CAR:

Availability
As of SpartaDOS X 4.41.

Remarks
This 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, and then hitting the RETURN key. To disable it type 'CON 40' and press RETURN.

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 KB of the main memory. This sets the MEMTOP value at $8035. Few programs are actually happy with this. Second, not every SpartaDOS X utility can cope with the screen configured so. For example, as of SpartaDOS X 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. See "Using CON: drivers in own programs" (Chapter 6).

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.

Notes: 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.

The CON64.SYS driver requires a XL/XE computer equipped with a 130XE-compatible memory extension.

CON64.SYS has been enhanced to work on machines with Axlon type extensions.

APPLICATION DRIVERS

RUNEXT.SYS Driver

Purpose

This provides file association support.

Syntax

DEVICE RUNEXT [d:][path][filename.ext]

Type

External - on device CAR:

Remarks

RUNEXT.SYS is an extension to the Command Processor, that allows to define associations between data 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 Control.

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
```

and hitting the RETURN key 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.

Testing of associations is doing 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.

COMEXE.SYS Driver

Purpose
Enables automatic cartridge management when launching programs.

Syntax
DEVICE COMEXE

Type
External - on device CAR:

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 SpartaDOS 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.

OTHER DRIVERS

ENV.SYS Driver

Purpose
Environment extension

Syntax
DEVICE ENV

Type
External - on device CAR:

Availability
As of SpartaDOS X 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 KB) 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.

This driver requires an ATARI computer with at least 2 banks of extended memory available.

DOSKEY.SYS Driver

Purpose

Command line extension.

Syntax

DEVICE DOSKEY [d:][path][fname.ext] [/X]

Type

External - on device CAR:

Availability

As of SpartaDOS X 4.42.
As of SpartaDOS X 4.45 DOSKEY handles filename completion.

Remarks

DOSKEY matured into a multipurpose tool being much more than the driver it used to be.

At first, if loaded, DOSKEY.SYS allocates one bank of extended memory (16 KB) to record commands the user types at the DOS prompt (BAT file commands are not recorded). The history buffer can hold up to 244 such command lines. After reaching this limit the oldest entries will be replaced with the new ones. Empty lines are not recorded. The commands can be invoked with the following key shortcuts:

- Control/P (as "Previous") or up arrow: invoke the previous (or older) command.
- Control/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.

DOSKEY will now clear the screen when you use the SHIFT/CLEAR key combination.

Note: when the history buffer is active, the user has only a limited set of control keys to edit the command line. Apart from the above, these are:

right arrow, left arrow, Backspace, Control/Insert, Control/Delete, Shift/Delete and (of course) Return.

At second, 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 '&'.

The third functionality offered is a possibility to define alternative names or aliases to DOS commands. They are defined in a text file. The name of this file should be given to DOSKEY.SYS as a parameter. The definition consist of the alias, the '=' sign, 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 KB, at least 1 KB (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.

The fourth brand new function is filename completion.: Pressing the Tab key while typing a command name will cause a search through COMMAND.COM commands, aliases and $PATH for matching command and file names. Pressing Shift/TAB tells the program to search also the hidden files (which are normally ignored). For example:

A:FS[Tab]

Since the only matching file is FSTRUCT.COM, this will cause the DOSKEY to supply this file name (i.e. "FSTRUCT.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.

Notes: Please remember that DOSKEY.SYS requires an ATARI computer with at least 128 KB of RAM.

DOSKEY used to crash when a line longer than 256 characters was typed. The length of the line is now limited to 64 characters (that's the supported amount).

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 MB)
E.	Total capacity (A*D):	503308800 bytes (~480 MB)
F.	Directory size:	32768 bytes (32 KB)
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 MB)
K.	Number of files open at a time:	16
L.	Path length:	64 characters
O.	Extended memory:	max. 2 MB

B Error Messages

Description of all error messages

The following is a list of error codes and messages that may occur while using SpartaDOS X. Some of the more common error codes are displayed in message form (as indicated by quotes) with most SpartaDOS X 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 (BASIC, ACTION!, etc.) errors and are not produced by SpartaDOS X.

The descriptions here are meant to cover the most common error conditions. It is possible but not likely to get some of these error codes or messages under different circumstances.

128 $80 "User break abort"

You pressed the BREAK key when the computer was waiting for input or printing to the screen. BREAK does not interrupt disk I/O in SpartaDOS X, but most programs will terminated after disk I/O is completed if the BREAK key has been pressed.

129 $81 "File already open"

You attempted to open a file for output that is already open. This can occur if you accidentally try to COPY a file on top of itself. For example, 'COPY MYFILE'.

Since the default destination is '*.*', this error will occur. No prior versions of SpartaDOS 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 SpartaDOS X 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"

You attempted to read from a file that was open for write only (mode 8 or 9). This error would indicate a programming error.

132 $84 "No device handler installed" (Bad CIO command)

You attempted 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 would indicate a programming error. You may also get this error when attempting to access a "kernel" device with no handler installed such as COM:.

133 $85 (File not open)

You attempted to perform a read or write (or note/point XIO operation) on a file that has not yet been opened. This indicates a programming error.

134 $86 "Bad file handle"

You called the CIO with an invalid IOCB number in the X register. You must multiply the IOCB number you wish to use by 16. This indicates a programming error.

135 $87 "File is read-only"

You attempted to write to a file that was open for read only (mode 4). This indicates a programming error.

136 $88 (End Of File)

This is 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. SpartaDOS X kernel calls return EOF status differently.

137 $89 "Truncated record"

This is not really an error but an indication that the record you attempted to read was longer than the buffer given to read the record into. This status may only be returned by an input function through the CIO. The SpartaDOS X kernel calls return this status differently.

138 $8A "Device does not respond"

You attempted 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: 1) Your disk drive door is open, 2) your Multi I/O is configured for a hard drive but none is online at that drive number, 3) you have a bad sector and your drive takes a long time to return any response, 4) you tried 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 your drive and computer are not communicating properly. If this error consistently happens, then you probably need your drive or computer serviced (It is possible that a serial framing error can occur if you have a bad sector, but it is unlikely).

141 $8D "Cursor out of range"

142 $8E "SIO overrun"

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 (It is possible that a serial bus overrun error can occur if you have a bad sector, but it is unlikely).

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 (It is possible that a serial checksum error can occur if you have a bad sector, but it is unlikely).

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 SpartaDOS 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"
SpartaDOS could not recognize the DOS format of the disk you attempted to access. If the diskette is ATARI DOS 2 format, then make sure that the ATARIDOS.SYS driver is installed in your system. It is installed by default if you do not have a CONFIG.SYS on drive 1. If you have 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"
You specified a directory path that does not exist. Recheck the pathname you specified. You may perform a directory command on each directory of your path to make sure that each directory exists.

151 $97 "File exists"
You attempted 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"
You attempted to LOAD or run a file that is not a binary load file. There are several scenarios in which this error can occur.

1) 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.)
2) You attempted to run a relocatable SpartaDOS X command file with the X command. The X command only loads standard ATARI binary load files.
3) 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 SpartaDOS X loader could not load a program because it accessed a symbol that has not been defined. This indicates that you need to first load the appropriate driver for the command you are attempting. 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 manual for command syntax and usage.

158 $9E "Out of memory"
You attempted to load or run a SpartaDOS X 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 in which there is not enough memory available is when you attempt to ARChive files on an unmodified ATARI 800 computer.

161 $A1 "Too many channels open"
SpartaDOS X 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, one 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 >>NEWCOM).

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 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"
Your disk is full. SpartaDOS X 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 file is removed from that disk.

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 MB for a regular file and 32 KB for a directory). Generally: a parameter for the operation is beyond the allowed limit.

167 $A7 "Directory not empty"

The directory you tried to delete contains files or subdirectories. You must 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. A subdirectory containing a "phantom" entry can not be deleted. You should use "CleanUp" from FTe's SpartaDOS Toolkit to remove this entry to allow the directory to be deleted. See more in Appendix E, "Using CleanUp on SpartaDOS X.

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. In earlier SpartaDOS X versions an attempt to exceed this limit caused the error 162 and the message "Disk full" used to appear, not quite accurate, since the files still can probably be saved to the disk, just not in this particular directory.

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.

179 $B3 "Memory conflict"

An attempt to load a program, which overlaps the DOS kernel or the I/O library area. It often means, 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.

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

C Command Summary – Alphabetical

Quick Reference List

It is intended as a quick reference for command syntax and usage. For more details concerning command operation please refer to chapter 4.

-fname [param1, param2 ,..., param9]

Executes the specified batch file, optionally passing parameters. The extension is assumed to be ".BAT" if none is specified.

APPEND *pathname*

Append the given 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.

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

Sets/clears file attributes in the directory. Replaces the Protect and Unprotect functions from older SpartaDOS versions. Alias – ATTRIB.

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

Enters the *internal* BASIC in a XL or XE computer (except 1200XL).

BLOAD [d:][path]fname[.ext] $address or address

Loads a file into the given memory area starting at the given address.

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

Tells a SpartaDOS formatted disk to load a specified file when the system is booted with this disk.

CAR [/I|N] [d:][path][frame] [parameters]

Enters the cartridge plugged into the top of the SpartaDOS X cartridge. If a filename is specified, then that binary file is loaded and run with the cartridge enabled.

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.

CHKDSK [d:] [/X|V]

Shows volume, free/total disk space, and sector size of the selected drive (or diskette).

CHTD [+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.

CHVOL [d:]volname

This command changes the volume name on the specified drive.

CLR

Deletes the environment variables, that were created by the system and are no longer used.

CLS [/F]

This command simply clears the screen. Especially for batch files it is very useful.

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.

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

Compares the given files.

CON 40|64|80

Enables and disables the 64- (CAR:) and 80-column (SDX TK) console mode.

COPY [/B|C|D|I|K|M|N|Q|R|S|V] [+A|H|P] [-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.

DATE [/T|dd-mm-yy or 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. Optionally, you can watch the directories with their containing files being deleted.

DEV

Display the list of available/installed kernel devices.

DF [/A]

Display summary information about free space on all disks.

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

Displays a long formatted directory with byte size, date and time.

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

Displays a short formatted directory (ATARI DOS 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.

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.

FIND [d:]fname[.ext]

Searches all directories on all drives for files matching the given filespec. If you enter a drive number, FIND will only look on that particular drive.

FMT [/S|J] fname[.ext] [ncol] or FMT [/S|J] <<fname[.ext]

A simple text formatter that reads the input line by line and applies the following to specified options.

FORMAT

Initializes any medium in SpartaDOS or ATARI DOS 2 format. A menu providing different options will appear.

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. See chapter 5.

INKEY

Stops batch processing and waits for a key press. Conditions may apply. See chapter 5.

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] fname[.ext] or LESS [/C] <<fname[.ext]

Text viewer with smart options. Converts MS-DOS and UNIX text files to ATARI format.

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

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.

MAN [/?] or **MAN [fname|/P]**

Starts the documentation viewer.

MAP [unit] [SIO|OS|NORMAL|OFF] [d:]

SIO.SYS control.

MDUMP $address $len or address len

Display memory in hex and ATASCII.

MEM [/X]

Displays the current low memory limits of your system, the available windowed RAM, the overall extended memory and its type.

MENU

Allows you to select files and then perform COPY, ERASE, RENAME, etc. commands on all selected files. It is similar to other SpartaDOS menu programs, but provides many new features.

MKDIR [d:]path

Creates a subdirectory. Alias - MD & CREDIR.

MORE <<fname.ext or 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 $address or address or PEEK symbol

Examines a memory location, performs a HEX conversion, or converts the given symbol to the associated address and memory index.

POKE $location $value or location value

Changes the content of a memory location.

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.

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 or address address

Saves binary data from memory to disk.

SET [var[=env_string]]

To display the values of all environment variables and optionally, sets an environment variable to a specified value.

SETPATHS [d:][path]|fname[.ext]

Invoked from a batch file the predefined paths are set on the addressed drives.

SIOSET [d: [type [usindex]]] or **SIOSET NMI [index]**

SIO.SYS serial speed control.

SL

This command generates a list of SpartaDOS X symbols.

SORTDIR [d:][path] [/N|T|S|D|X]

To sort filenames in directories by name, type, date or size.

SWAP [d,d]

To swap (Re-map) your drive configuration or to 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 you 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).

VER

Displays the current version number and date of the cartridge.

VERIFY ON|OFF

Turns write verify on or off.

X [/C] [d:][path]fname[.ext] [parameters]

Executes a program which requires that no cartridges are installed (such as "DiskRx", EXPRESS, most binary files, etc.).

D Command Summary – By Function

This list is intended as a quick reference for command syntax and usage. For more details concerning operation please refer to chapter 4 and 5.

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.

PAUSE
Suspends system processing and displays the message "Press RETURN to continue". Optionally accepts a number of seconds to wait, ranged from 0 to 65535.

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.

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. See chapter 5.

INKEY
Stops batch processing and waits for a key press. Conditions may apply. See chapter 5.

PROC
Starts a procedure in a batch file. See Chapter 5.

RETURN

Ends a process in a batch file. See Chapter 5.

EXIT [n]

Causes the immediate termination of a processing batch file. Optional exit code to be taken by the system as error code.

SETPATHS [d:][path]|fname[.ext]

Invoked from a batch file the predefined paths are set on the addressed drives.

Directory Commands

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

Displays a long formatted directory with byte size, date and time.

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

Displays a short formatted directory (ATARI DOS type).

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.

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.

DELTREE [/YV] [d:] path

Delete subdirectory trees recursively. Optionally, you can watch the directories with their containing files being deleted.

SETPATHS [d:][path]|fname[.ext]

Invoked from a batch file the predefined paths are set on the addressed drives.

MENU

Allows you to select files and then perform COPY, ERASE, RENAME, etc. commands on all selected files. It is similar to other SpartaDOS menu programs, but provides many new features.

SORTDIR [d:][path] [/N|T|S|D|X]

To sort filenames in directories by name, type, date or size.

Disk Maintenance Commands

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

Tells a SpartaDOS formatted disk to load a specified file when the system is booted with this disk.

CHKDSK [d:] [/X|V]

Shows volume, free/total disk space and sector size of the selected drive (or diskette).

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

Initializes a disk in SpartaDOS or ATARI DOS 2 format. A menu providing different options will appear.

VERIFY ON|OFF

Turns write verify on or off.

MAP [unit] [SIO|OS|NORMAL|OFF] [d:]

SIO.SYS control.

PWD

Outputs a list of current working directories.

SIOSET [d: [type [usindex]]] or **SIOSET NMI [index]**

SIO.SYS serial speed control.

File Maintenance Commands

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

Sets/clears file attributes in the directory. Replaces the Protect and Unprotect functions from older SpartaDOS versions. Alias – ATTRIB.

BLOAD [d:][path]fname[.ext] $address or address

Loads the given file into the given memory area starting at the given address.

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

Compares the given files

COPY [/B|C|D|I|K|M|N|Q|R|S|V] [+A|H|P] [-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.

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).

MENU

Allows you to select files and then perform COPY, ERASE, RENAME, etc. commands on all selected files. It is similar to other SpartaDOS menu programs, but provides many new features.

Running Programs

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

Enters the *internal* BASIC in a XL or XE computer (except 1200XL).

CAR [/I|N] [d:][path][frame] [parameters]

Enters the cartridge plugged into the top of the SpartaDOS X cartridge. If a filename is specified, that binary file is loaded and run with the cartridge enabled.

X [/C] [d:][path]fname[.ext] [parameters]

Executes a program that requires no cartridges to be installed (such as "DiskRx", EXPRESS, most binary files, etc.).

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]

To swap (Re-map) drives or to display the current drive map list.

Time - Date Support

CHTD [+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 or mm-dd-yy]

Displays the current date and allows you to set the date. DAYTIME settings apply here.

TIME [/T|hh:mm:ss]

Displays the current time and allows you to set the time.

TD ON|OFF

Turns a time/date display line on top of your screen on and off.

Utilities And Programming Aids

APPEND *pathname*

Append the given 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- (CAR:) and 80-column (SDX TK) console mode.

DEV

Display the list of available/installed kernel devices.

DUMP [d:][path]fname[.ext] [start] [len]

Displays a file in HEX and ATASCII form.

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

Enable text editor.

FIND [d:]fname[.ext]

Searches all directories on all drives for files matching the given filespec. If you enter a drive number, FIND will only look on that particular drive.

FMT [/S|J] fname[.ext] [ncol] or FMT [/S|J] <<fname[.ext]

A simple text formatter that reads the input line by line and applies the following to specified options.

LESS [/C] fname[.ext] or LESS [/C] <<fname[.ext]

Text viewer with smart options. Converts MS-DOS and UNIX text files to ATARI format.

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

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.

MAN [/?] or **MAN [fname|/P]**

Starts the documentation viewer.

MDUMP $address $len or address len

Display memory in hex and ATASCII.

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 $address or address or PEEK symbol

Examines a memory location, performs a HEX conversion, or converts the given symbol to the associated address and memory index.

POKE $location $value or location value

Changes the content of a memory location.

SAVE [d:][path]fname[.ext] $address $address or address address

Saves binary data from memory to disk.

TYPE [+A|H|P|S] [-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

Using Turbo BASIC XL with SpartaDOS X

The well-known Turbo-BASIC XL (TBS XL) uses the RAM under the OS ROM in XL/XE computers. Therefore it is not compatible with SpartaDOS 2.x and 3.x. However, with the proper hardware and configuration TBS XL will work well with SpartaDOS X (SDX).

Hardware Configuration

A XL/XE computer with more than 64 KB is needed to use TBS XL with SDX. The 400/800 computers do not have RAM under the OS. XL/XE computers with 64 KB memory or less are also unsuitable, since SDX needs to acquire the RAM under the OS on these machines (USE OSRAM). This conflicts with TBS XL using the same areas.

System Configuration

One bank of extended memory (USE BANKED) is needed to run TBS XL. If you have more than 128 KB of RAM in your computer, this will occur by default. If you have just 128 KB however, it is required that you boot with a custom CONFIG.SYS file from a SpartaDOS formatted medium to be able to use TBS XL. See chapter 8 about the "config selector" for a possible alternative.

The first line of any CONFIG.SYS while using TBS XL *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 KB and TBS XL:

```
USE BANKED
DEVICE SPARTA
DEVICE SIO
DEVICE JIFFY (or a driver for your RTC)
DEVICE RAMDISK O,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 TBS XL, 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 SpartaDOS X.

Using AUTORUN.SYS files

SpartaDOS X, when booted, will not automatically load and run a file named AUTORUN.SYS. With batch files and the relocatable nature of the SpartaDOS X 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.

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.

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.

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
```

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.

Using BASIC XE with SpartaDOS X

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 KB or 128 KB 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
```

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 SpartaDOS X. Fortunately, these problems can be easily avoided.

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, so the statement expressed in the original SpartaDOS X (4.2x) Reference Manual, that you can use BASIC XE Extensions in OSRAM mode is no longer valid. If you want to load the BASIC XE Extensions, the DOS must be configured in BANKED mode and the computer has to have more than 128 KB of RAM.

Other Conflicts

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 with the COLD command to wipe out the extensions before running other programs.

Using MAC/65 and DDT with SpartaDOS X

MAC/65 works well with SpartaDOS X, 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 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 '/I' switch (the same applies to other OSS-type cartridges).

Using AtariWriter Plus with SpartaDOS X

With a stock 130XE or 800XL computer, using AtariWriter Plus is straightforward. Simply insert the AtariWriter Plus diskette into D1: and type 'D1:**X AP.OBJ**'. If you have more than 128 KB of RAM in your computer, the procedure is a bit more complex. You will need to prepare a boot floppy disk for AtariWriter Plus. FORMAT a disk in SpartaDOS format and create a text file named CONFIG.SYS file on it. These lines must be in the. CONFIG.SYS:

```
USE OSRAM
DEVICE SPARTA OSRAM
DEVICE SIO
DEVICE ATARIDOS.SYS
```

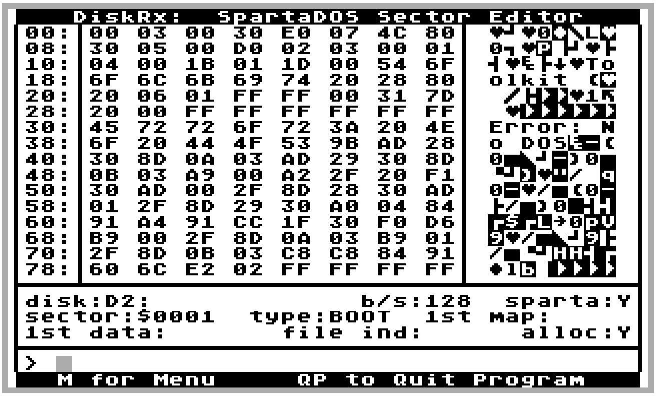
You may use the rest of this disk for anything you choose. To run AtariWriter Plus, boot the computer with this boot floppy disk in D1:. Remove this diskette and insert the AtariWriter Plus diskette and then type 'D1:**X AP.OBJ**'.

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.

The above hints for AtariWriter Plus have been kept for historical reasons in the manual and they still apply. Today’s word processors are much more powerful. It is recommended to check 'The Last Word' available from http://atari8.co.uk/ as it especially supports SpartaDOS X features.

Disk Editors to use with SpartaDOS X

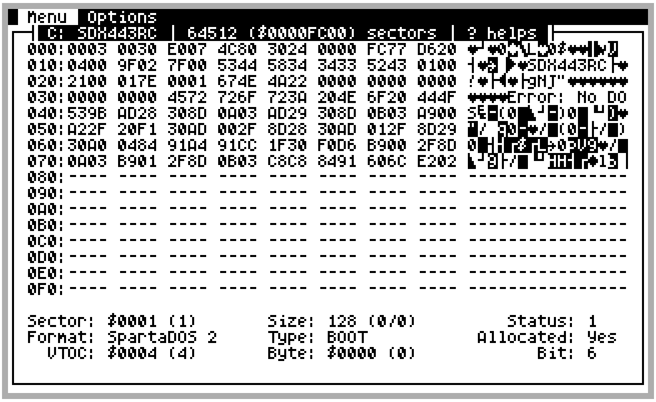
FTe's "DiskRx" is the classic sector editor to be used with SpartaDOS X. Unfortunately, DiskRx does not recognize the new DD 512 density.



```
00:  00 03 00 30 E0 07 4C 80
08:  30 05 00 D0 02 03 00 01
10:  04 00 1B 01 1D 00 54 6F
18:  6F 6C 6B 69 74 20 28 80
20:  20 06 01 FF FF 00 31 7D
28:  20 00 FF FF FF FF FF FF
30:  45 72 72 6F 72 3A 20 4E
38:  6F 20 44 4F 53 9B AD 28
40:  30 8D 0A 03 AD 29 30 8D
48:  0B 03 A9 00 A2 2F 20 F1
50:  30 AD 00 2F 8D 28 30 AD
58:  01 2F 8D 29 30 A0 04 84
60:  91 A4 91 CC 1F 30 F0 D6
68:  B9 00 2F 8D 0A 03 B9 01
70:  2F 8D 0B 03 C8 C8 84 91
78:  60 6C E2 02 FF FF FF FF

disk:D2:                 b/s:128   sparta:Y
sector:$0001  type:BOOT   1st map:
1st data:          file ind:         alloc:Y

>
```

Fig. 43: Edit screen DiskRx 1.9

Another problem with "DiskRx" is that the program checks the file system type on the medium from which it is loaded and refuses to work, if it is not a SpartaDOS file system. Since the source code is not available, there was no other solution for that, than to write a suitable media editor from scratch. The program has been written. It is called 'Eddy' and is part of the SpartaDOS X Toolkit.

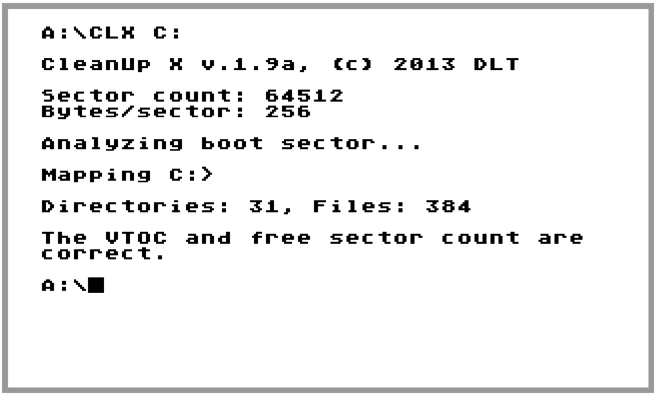


```
C: SDX443RC | 64512 ($0000FC00) sectors

000:0003 0030 E007 4C80 3024 0000 FC77 D620
010:0400 9F02 7F00 5344 5834 3433 5243 0100
020:2100 017E 0001 674E 4A22 0000 0000 0000
030:0000 0000 4572 726F 723A 204E 6F20 444F
040:539B AD28 308D 0A03 AD29 308D 0B03 A900
050:A22F 20F1 30AD 002F 8D28 30AD 012F 8D29
060:30A0 0484 91A4 91CC 1F30 F0D6 B900 2F8D
070:0A03 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: BOOT          Allocated: Yes
  VTOC: $0004 (4)       Byte: $0000 (0)           Bit: 6
```

Fig. 44: Edit screen Eddy 2.01

File system consistency checkers for SpartaDOS X

The "CleanUp" program from FTE's SpartaDOS Toolkit is a powerful tool, which unfortunately has some bugs and cannot handle the newly introduced DD 512 density. Therefore it is strongly recommended to use the new tool not only when using 512 DD.



```
A:\CLX C:

CleanUp X v.1.9a, (c) 2013 DLT

Sector count: 64512
Bytes/sector: 256

Analyzing boot sector...

Mapping C:>

Directories: 31, Files: 384

The VTOC and free sector count are
correct.

A:\█
```

Fig. 45: CLX 1.9a checking DD medium

The current version of "CleanUp X", the new file system consistency checker for all versions of SpartaDOS, is part of the SpartaDOS X Toolkit and maybe downloaded from

http://sdx.atari8.info/ .

F System Drivers Summary

Alphabetical quick reference

ARCLOCK.SYS [/F]
The driver for the battery backed "ATARI Real Clock".

ATARIDOS.SYS
Reads and writes ATARI DOS 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
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 SpartaDOS kernel and the main filesystem driver.

XEP80.SYS [1|2] [/PN]
The XEP80 driver.

Z.SYS [/I|S]
Installs a SpartaDOS 3.2-compatible 'Z:' device.

G Enhancements And New Features

Please find here an overview of the latest changes and unsolved issues. These point to the current and future developments of SpartaDOS X. The complexity of nowadays hardware and all the features asked for by the users of SpartaDOS X lead to a continuing process of enhancement and development. These changes relate to version 4.45.

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.

Kernel

- CONFIG's keyword MERGE now allows merging portions of CONFIG.SYS from arbitrary drive/path. Details:

  The merged config file will be read using SDX device drivers, not OS routines (as before). This allows drives/partitions that are accessible only via SDX drivers (SIDE, MyIDE etc.) to hold the merged configuration part for easy change. The main config should contain USE command (memory configuration), device drivers (such as DEVICE SIO, DEVICE SIDE) and then "MERGE filename.cfg" to attach the device-resident part of the configuration.

  Additionally, in case the filename contains either a device name, a drive number or a path, the merged config will be loaded using SDX filesystem drivers (such as SPARTA.SYS). This way a file from a different directory/filesystem can be merged (for example MERGE D2:>CONFIG>XTRA.CFG). Note that this requires the SDX filesystem driver to be loaded earlier (DEVICE SPARTA). Also beware that the merged files cannot exceed 1KB.

  If just the filename is given (like in previous SDX versions), the filesystem drivers are not used to get the merged config. In such case the merged file must be located in the same directory as the file that merges. If the file is merged from the default CONFIG.SYS on CAR:, it should be stored in the root directory of the boot drive.

  If no file name is given (MERGE without a parameter), SDX config selector is invoked to let you choose from multiple CFG files stored in SPARTA.DOS directory (see the manual for details). Note that in case of OS-compliant boot drive, the config selector is shown regardless of MERGE command.

  If a user hold OPTION key during system startup, no files are merged.

- Expanded support for standard RAM extensions: internal variables now not only hold the current number of free extended banks, but also the default one. Thanks to that, MEM can now display the information how much ext RAM is there in general (not only, what is free, as before).
- Added support for 65C816 high RAM: when 65C816 is detected, the DOS will now do a non-destructive sizing of that memory at reset time and store the following information on that: the starting segment number (such as $01 when the high RAM starts at $010000), and the number of extra segments besides the segment 0 (i.e. besides the 6502 conventional 64k space). Both numbers are 0 when nothing is detected. Again, MEM will now use this information, if applicable.
- Added experimental support for an Axlon-type extension on XL/XE computers. The controlling register should be at $CFFF (write-only), and should only be active when PB0=1. Unlike on Atari 800, there is no shadow assumed at $0FC0. The extension is only tested for when the test for a standard type extension returns 0 banks. Up to 127 extra banks (2032 KB) can be detected. When Axlon is detected, only two memory configurations are possible: USE NONE or USE BANKED. USE OSRAM is ignored.
- MEMLO somewhat (around ~40 bytes) lowered.

Library

- S_NEXT, new symbol to walk through symbol list.
- S_ADD now does not add a symbol, if there is not enough memory left to create one.
- U_LOAD (and its equivalent located inside X.COM) will now store an absolute path in 'path' ($07a0) to the directory where the executable is residing. Therefore a program will be able to easily retrieve its path upon execution (provided that no other I/O was done in meantime). This seems to fix the problem with MyDUP requiring COMEXE.SYS to be resident: it should not require that anymore.
- SLEEP, new symbol to make precise delays.
- BLDDIR_P, new symbol to build directories overriding the device-returned PERCOM block.
- BUILDDIR will now correctly mark double sided disks as double sided.

Drivers and resident programs

- The serial I/O driver will now always have the lowest possible priority among all the other SIO alike drivers (ramdisks, SIDE/MyIDE drivers and such), regardless of the order of loading.
- Directory formatting routines, when writing the final "FREE SECTORS" line to the output buffer, will now zero out the status byte in the directory buffer. This enables programs which read the formatted direct-

ory byte by byte to detect the final line of the directory listing before its is completely read out.

- INDUS.SYS did not work, fixed. Also, in 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 on higher drives, edit the CAR:CONFIG.SYS and change or remove the parameter digit in DEVICE INDUS line.
- File system drivers modified to return DOS 2.0 "status $03". The effect is that Turbo BASIC XL "BLOAD" command now works.
- RAMDISK.SYS has a new switch. '/S' forces loading the "standard" 6502 RAM driver module on 65816 machines. The 6502 module is remarkably slower, but has the advantage of occupying much less memory than 65816 mods.
- CON64.SYS now works on computers with Axlon extension.
- ENV.SYS now works on computers with Axlon extension.
- ATARIDOS.SYS will now mark zero-length files in the directory as occupying 1 sector, not 0 sectors as before.

Utilities

- COPY: when a file being copied has no time stamp, current time and date gets assigned to it.
- MEM /X will now display the total amount of extended RAM in banks and kilobytes, besides the free amount.
- MEM /X also displays the amount of 65C816 high RAM, if applicable.
- ED will not quit after ESC/RETURN, if saving the file fails. Also SHIFT+DELETE will set modification flag.
- X.COM experimentally shortened by moving a portion of code to the cart.
- CHKDSK /X display underwent some cosmetic changes. Plus some bugs fixed related to correct distinctions between physical sectors and logical clusters.
- DATE and TIME syntax is now "DATE [/T|dd-mm-yy]" and "TIME [/T|hh:mm:ss]", respectively. When used with /T parameter, will now only display current date or time and will not prompt for entering new values. When fed with a valid date/time value in the command line, it will set the specified value as current.
- PAUSE now optionally accepts a number of seconds to wait, ranged from 0 to 65535.
- TYPE /P and MORE should work better when displaying text files containing long lines.
- ARC will now no longer ask if to overwrite a file when unpacking an archive to NUL:
- ARC is now able to fetch files to be archived from any file-oriented device, not only from regular disks (DSK:) as before.

- DELTREE has now more MS-like syntax: DELTREE [/YV] [d:]path. The /V switch enables a "verbose" mode, which allows to watch what files are currently being deleted.
- DELTREE now displays complete pathnames of the directories being deleted, and files too, if asked to (see above).
- LESS has a new function: when you press the 'G' key, you'll be prompted for a line number the viewer should jump to.
- LESS now automatically detects MS-DOS (CP/M) and Unix line endings and converts the text accordingly on the fly. Therefore, you now may use the command to convert MS-DOS and Unix 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. Only EOL and TAB characters will be converted, but that's enough most of the time.

Things done and still to do

To give you an idea of the ongoing development process and provide some insights please find information about known bugs, that have been fixed as well as information about known bugs being on the list for fixing.

Bugs fixed

- RAMDISK.SYS fixed to work with Axlon extension (this was broken, apologies).
- 16kbyte cartridges (Microsoft Basic II, Logo, etc.) will occupy memory from $8000 only when they are active, i.e. when CAR command is invoked. Such cartridges caused a lock in SDX 4.4x or reserved the area upon bootup even when they were inactive (SDX 4.2x).
- ED will not quit after ESC/RETURN, if saving the file fails. Also SHIFT+DELETE will set modification flag.
- MENU should no longer accept random garbage it finds in the directory buffer, when it fails to load the directory selected.
- LESS now expands TABs rather than converting each one to a single space.

To-Do List

- ARC A has sometimes a problem packing very long files (confirmed).
- ACTION! monitor doesn't display commands if QUICKED is loaded.
- CON.SYS has a problem handling lines longer than 80 characters: it stops reading the text from the screen, as if there was nothing typed in the line.
- 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.
- 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.

Table of Figures

Screenshots

Index

In alphabetical order

SpartaDOS X

v. 4.46

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 KB of memory, the all new SpartaDOS X supports all modern devices and gadgets available for the ATARI 8-Bit line of computers. And it is a living project.

This manual provides you with an in-depth look into the sophisticated world of the new SpartaDOS X, beginning with the basics of DOS to a technical analysis of the disk and file structure utilizing hard drive partitions up to 32 MB 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 POKEY divisor 0, 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 the new SpartaDOS X cater to the entire 8-bit line, but also employs full use of extended memory up to 2 MB to be used 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 all new SDX Imager and the on-line help system will support your usage a lot in getting more out of the system while using the new SpartaDOS X.

DLT, June 2013

SDX
4.46

SpartaDOS X

User's Manual

Jun
2013