Version 4.42
December 2008

SpartaDOS X

Features & Support

- Turbo Freezer
- Atrax SDX 128
- MaxFlash Cart
- XEP80 for PAL Computers
- 15 Drives
- 512 Byte Sectors
- 32 MB Partitions
- Updated Kernel
- Online Help System

International ATARI 8-Bit Community

We are proud to present the enhanced and remarkably upgraded

SpartaDOS X version 4.42

The most advanced Operating System
for ATARI 8-Bit Computers

SpartaDOS X on cartridge was released end of 1988. This powerful OS was said to be enhanced and developed. Changes in the computer world in the 1990s made it to an abandoned product. Last known version from those days is SpartaDOS X V. 4.22 from FTe.

As of December 2008 there are new features and capabilities available:
Supporting *Turbo Freezer 2005 cartridge*, *AtraX SpartaDOS X 128* and *Atarimax Maxflash cartridge.* The banked memory management has been updated and can handle all known memory extensions properly. A new file system has been written to support more storage devices and 512 bytes per sector. 15 drives are available with SpartaDOS X. The Y2K-bug is fixed. Proper support for the XEP80 video card on PAL computers is implemented. Utilities known from the SpartaDOS Toolkit have been incorporated. Time and date inputs entered by the user are verified.
And there is much more you will explore reading the manual and using SpartaDOS X.
And, of course, the new SIO2XX devices are supported as well.

See *http://sdx.atari8.info/.* for more technical details.

Please take this manual for courtesy.

Keep ATARI 8 bit computers alive!

Enjoy!

Christmas 2008

CREDITS

- based on works done by: Prof!, MMMG, DLT Ltd.
- new code and design: DLT Ltd.
- hardware: Pasiu/SSG, Jad, Zenon/Dial, DLT Ltd.
- hosting: krap.pl
- devtools: DLT Ltd., Tebe/Madteam, others
- manual: Mikey, dely, DLT Ltd, GoodByteXL
- other support: ABBUC, Epi/TRS, Krap, Mikey, Pin/TRS

SpartaDOS X

Version 4.42

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.
The ATARI users community considers these products to be abandonware.

Atari 130 XE, 800XL, 400/800, 810, 850, 1050, XE Game System, XEGS, 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.
Axlon still existed as a royalty-collecting company in 1999 – Current status unknown.

MSDOS is a trademark of Microsoft Corporation.

UNIX is a trademark of AT&T.

ARC is a trademark of System Enhancement Associates.

First version published by ICD, Inc., 1988.
Last commercial version by FTe, 1995.

New Enhanced Version by DLT Ltd. 2008.

SpartaDOS X is an abandoned product for more than 10 years.

Enhanced and kept alive by the ATARI 8-bit community.

Table of Contents

1 Introduction

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 operates and how you can take advantage of its power.

Chapter 3 describes the general operation of SpartaDOS X and the ways in which it differs from previous versions of SpartaDOS. You will also find the newest information about the latest changes and bugfixes for the current version here. Every user should read this chapter.

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 his or her own programs.

Chapter 7 follows with a technical and detailed look at SpartaDOS X disk and directory structure. This chapter will probably not be of interest to most 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 computer's modifications (such as extended memory) and peripherals to provide maximum flexibility and control.
The appendices cover such topics as error messages, command summaries, and common problems and solutions.

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 product, we advise you to post respective questions on supporting ATARI 8-bit fora on the Internet.

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.

Technical Advice: The full potential of SpartaDOS X is only available 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 - The more, the better ...

Note: If you are using a different device than the new designed 128 KB SpartaDOS X cartridge or the internal solution, as there are e.g. AtariMax FlashCart or TurboFreezer or even an Emulator, please be aware of the limitations with it. For example it is not possible to put language cartridges like e.g. ACTION!, BASIC XL or MAC/65 into a virtual SpartaDOS X cartridge. Maybe future versions will provide a solution for it.

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, in a tutorial format, several examples of elementary SpartaDOS operations. If you read through the chapter and perform all of the examples, taking time as necessary to experiment, you should be well on your way to becoming a SpartaDOS power user.

The Command Processor

SpartaDOS differs from Atari DOS 2 and clones 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 printed on the screen and may be selected with a single letter. SpartaDOS uses a command processor (CP); at a prompt, the full command is typed. Those of you familiar with MSDOS on IBM personal computers and clones, 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 much more powerful and flexible. A menu is also included to make file maintenance, especially with multiple files, very quick and easy.

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 have a 400 or 800 with 48 KB or any XL/XE computer with 64 KB of RAM, you will see the following message:

```
RAMDISK not installed
No extended RAM
```

If you have a stock 130XE (128 KB of RAM) or any other XL/XE computer with 128 KB of RAM, you will see this message:

```
Ramdisk not installed
64k reserved for XE programs
```

You will then get a message giving the version number and date of your SpartaDOS X cartridge and a copyright notice,

```
SpartaDOS X 4.42 25-07-2008
Copyright (c) 2008 by FTE & DLT
```

with the current revision number and the current revision date followed by a prompt:

```
D1:
```

Formatting a Disk

Before you can begin to explore the command processor, you need to have a floppy disk with which to experiment. Place a new disk in drive #1 and type

```
D1: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 that is in bold face. You should press RETURN after each line typed unless instructed otherwise.
After typing FORMAT and pressing the RETURN key, you will see the SpartaDOS formatter menu. Press the 1 key to select drive #1. Press the V key, type TESTDISK, and press the RETURN key. 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 described in chapter 4 under the FORMAT command. You can just ignore them for this exercise.

Disk Directory

After the format is completed, press the ESC (escape) key to return to the D1: prompt. Type

```
D1:DIR
```

You should now see something similar to

```
Volume:    TESTDISK
Directory: MAIN

   714 FREE SECTORS
```

The actually free sector count may vary depending upon the type of disk drive you are using.

Creating Test Files

The disk has no files on it now. To be able to experiment with commands, we need some files on the disk. If you have an XL or XE computer (except for the 1200XL) type

```
D1:BASIC
```

If you have 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 cotton swab, pencil eraser, or other small item into the hole located on the front right edge of the cartridge area.

Type

```
D1:CAR
```

Regardless of computer type, you should now see the BASIC prompt:

```
READY
```

Type in this short BASIC program:

```
10 OPEN#1,8,0,"D1:TEST.DAT"
```

```
20 FOR X=0 TO 255
30 PUT#1,X
40 NEXT X
50 CLOSE#1
```

Type in

```
SAVE "D1:TEST.BAS"
```

to save the program and then type

```
RUN
```

to execute it. At the READY prompt return to DOS by typing

```
DOS
```

Now type

```
D1:DIR
```

You will now see

```
Volume:    TESTDISK
Directory: MAIN

TEST     BAS    147 25-07-08 10:49
TEST     DAT    256 25-07-08 10:49
   710 FREE SECTORS
```

As you can see, this directory format is different from that produced by any other DOS. The first one to eight characters (four in this case) are the *filename.* The next zero to 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.

Setting the Time and Date

If you do not have a R-Time 8 or ARC cartridge you will need to manually set the time and date. This will allow you to keep track of when your files were created and which is the latest version of a particular file. To do this, type

```
D1:DATE
```

You will see something similar to

```
Current date is 25-07-08
Enter new date (DD-MM-YY):
```

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. Now type

```
D1:TIME
```

You will see

```
Current time is 10:55:32
Enter new time (HH:MM:SS):
```

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). The time and date are set. Now type

```
D1:CHTD TEST.BAS
```

This will **CH**ange the **T**ime and **D**ate of "TEST.BAS" to the current time and date. Now type

```
D1:DIR
```

You will see that the file "TEST.BAS" now has the current time and date. Now type

```
D1:TD ON
```

This turns the Time and Date display ON at the top of the screen. This display may be disabled by typing

```
D1:TD OFF
```

Parameters

Many commands require parameters. In the above example for the command CHTD, the filename "TEST.BAS" part was a parameter. With

TD, "ON" and "OFF" were 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 for 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

```
D1:COPY TEST.BAS MAKEDAT.BAS
```

and list the directory (with the DIR command). You will 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 must use the MENU command (see chapter 4).

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

```
D1: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 and most 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 properly explore wildcards you will need some more files on your disk. Type in the following lines:

```
D1:COPY TEST.DAT ABCDE.DAT
D1:COPY TEST.DAT ABZDE.DAT
D1:COPY TEST.DAT ABCRAIG.DAT
D1:COPY TEST.DAT TEST.DOG
D1:COPY TEST.DAT TEST.DZT
D1:COPY TEST.DAT ABCDE.ICD
```

Earlier it was mentioned that some commands have default parameters if none are provided. DIR is one such command. The proper syntax is

```
DIR [fname.ext]
```

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 the directory. Type

```
D1:DIR TEST.DAT
```

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

```
D1:DIR *.DAT
```

Several files are listed, but only those with an extension of "DAT". Now try

```
D1:DIR A*.D*
```

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

Now try

```
D1: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.

Wildcards should be used with care with the ERASE and RENAME commands, since they can be used to 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. Within the "PICTURES" subdirectory above you could create a "WILDLIFE" directory and a "CARTOONS" directory, for example, and place the appropriate files in these.

Type in

```
D1:MKDIR TESTS
```

This means **M**a**K**e **DIR**ectory TESTS. Now list the directory. You will see an entry that looks like

```
TESTS        <DIR>  11-10-08 19:59
```

As always, your actual time and date will be different. You have just created a subdirectory, or "folder", named "TESTS". It is, however, empty. To verify this, type

```
D1:DIR TESTS>
```

The '>' character signifies that the preceding name was 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

```
Directory: MAIN
```

has now been replace by

```
Directory: TESTS
```

to show 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.

COPY is another command that allows wildcards. The following line will copy all of the files ending in ".DAT" from the main directory to the subdirectory TESTS:

```
D1:COPY *.DAT TESTS>*.*
```

Now get rid of the .DAT files in the main directory:

```
D1:ERASE *.DAT
```

List the directory to assure yourself that the ".DAT" file 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

```
D1:CHDIR TESTS
```

This means *CH*ange *DIR*ectory. Now type

```
D1: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:MKDIR ANOTHER
D1:CHDIR ANOTHER
D1:CHDIR
```

The last CHDIR with no arguments will show the path from the main directory to the current directory. In this case you will see

```
>TESTS>ANOTHER
```

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:

```
D1: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:

```
D1:CHDIR ..
```

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

Running Programs

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

```
D1: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

```
D1:DEMO.
```

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 600XL, 800XL, and 65XE, 130XE, or the XEGS type

```
D1:BASIC
```

and to enter an external cartridge (such as ACTION!, MAC/65, BASIC XL, BASIC XE, etc.) type

```
D1: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

```
D1:X DISKRX  or  D1:#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 – Configuring Your System 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,

```
D1:-TEST
```

would execute the batch file "TEST.BAT", while

```
D1:-DO_IT.TXT
```

would execute the batch file "DO_IT.TXT".

SpartaDOS X will automatically execute a batch file on D1: called "AUTOEXEC.BAT" when booted if one exists. This allows you to have several commands executed every time you boot the computer. For example, suppose that you do not have an R-Time 8 or ARC cartridge, but you still wish to install the time/date line at the top of the screen and set the proper values. Type the following line:

```
D1:COPY CON: D1:AUTOEXEC.BAT
```

What this will do is copy from the CON: device (the screen editor) to a file named "AUTOEXEC.BAT". The cursor will move to the start of the next line on the screen. Type in the following, ending each line with a RETURN:

```
TD ON
TIME
DATE
```

After you have entered the last line (and followed it with a RETURN), press CONTROL+3 (also referred to as ^3). To do this, press and hold the 'CONTROL' key. While holding down the control key, press the '3' key. This will signal the computer that the end of the CON: "file" has been reached. List the directory. You should now have a file called "AUTOEXEC.BAT" on D1:.
Type now the command

```
COLD
```

and press RETURN. The computer will reboot now and you should see the commands being executed from the batch file. Enter the TIME and DATE as before.

Batch files are one of the most useful features of SpartaDOS X. You can create them as above or with any word processor or editor that will save a file as straight text (without formatting commands). See also "ED Command" and "COLD 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**
N - CREATE MEM.SAV	**SET CAR** and **SET BASIC**
O - DUPLICATE FILE	**MENU**

SpartaDOS X contains numerous commands that have no equivalent in Atari DOS 2.0s. It also supports floppy drives of all sizes in single, dual, and double density, hard drives, ramdisks, SIO2XX devices, time/date stamping, subdirectories, and more.

3 SpartaDOS X Overview

This chapter is an overview of SpartaDOS X 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 a 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 and from the good old days of homecomputing. So utilizing extended memory for drivers from a 512 KB SRAM extension is a no frills job.

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 ('_')

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 SpartaDOS batch file
.BXL      A BASIC XL SAVEd program
.BXE      A BASIC XE SAVEd program
.COM      A SpartaDOS external command or binary program
.DAT      A data file
.DOS      A disk-based version SpartaDOS
.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 SpartaDOS system file or driver
.TUR      or '.TBS' is a TURBO BASIC SAVEd program
.TXT      An ASCII or ATASCII text file
```

In some cases the extensions will be assumed by the command processor or application. For example, ".COM" is assumed for command programs, ".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. For example

```
DIR AB?DE.XYZ
```

lists all directory entries on the default drive with filenames that have five characters, begin with "AB", have any next character, have "DE" for the next two characters, and have an extension of "XYZ", such as

```
ABCDE.XYZ
AB_DE.XYZ
ABZDE.XYZ
```

and similar file names in the current directory.
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. For example,

```
DIR AB*.XYZ
```

lists all directory entries on the default drive with filenames that begin with "AB" and have extensions of "XYZ", such as

```
ABCDE.XYZ
ABCRAIG.XYZ
AB.XYZ
```

and similar entries. It is important to note that any characters *after* the asterisk in either the filename or extension will be ignored, so that

```
DIR AB*DE.X*Z
```

will list the following directory entries, assuming that they exist:

```
ABCDE.XYZ
ABCRAIG.XXX
AB.X
```

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.)

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 200 or so. 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. 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 default 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.

Device Identifiers

SpartaDOS device identifiers in earlier versions were to be used through BASIC or anywhere else in the system - all 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 above are just a few.)

So, here is what we have come up with. Through 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 default drive or drive 1 - it is drive 4

: Since there is no drive letter, this is the default 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 *default 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 above examples showed 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 CWD). For example

```
CHDIR DOS
```

sets the current directory or the default drive to "DOS", and the command

```
CHDIR B:BASIC
```

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

Volume Names

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

- To better organize your diskettes by naming them.

- For SpartaDOS to quickly tell the difference between 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 an Atari DOS 2 format diskette, you will find that they are always named "DOS 2.0" so that you can quickly tell the difference between diskette formats. Also, because there is no unique volume name on Atari DOS diskettes, the buffer system does not remember anything about the last access to these diskettes. Therefore, Atari DOS diskettes are much less efficient and are slower. In addition, loading binary files, especially those with many segments, can take *considerably* longer from an Atari DOS disk than from a SpartaDOS disk, sometimes many minutes more. It is recommended that files be copied to SpartaDOS disks or ramdisks before running.

Disk Format Compatibility

SpartaDOS X 4.4x has no problem reading from and writing to disks formatted and used with SpartaDOS 2.x, 3.x and 4.x. It can also read a disk formatted with SpartaDOS 1.1. It is, however, not recommended to do any write operations (including file deletions) on SpartaDOS 1.1 diskettes, should they remain accessible to SpartaDOS 1.1 afterwards. The advice is to copy the files over to a SpartaDOS X disk first, and edit thereafter.

SpartaDOS X 4.4x is able to build file systems on disks formatted to 512 bytes per sector. This density, new to the Atari world, is called DD 512. Files saved on such a disk 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 also have no problem reading from or writing to disks formatted and/or written to by SpartaDOS X with these exceptions: any 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 disks are not compatible with any other SpartaDOS before V 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 Multi I/O, you may plug the SpartaDOS X cartridge into either of the slots on the connector.

If you have an 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 BASIC XL, BASIC XE, ACTION!, and MAC/65) 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 on demand. 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: Procedures with cartridges are different running SpartaDOS X from virtual cartridges like TurboFreezer etc.

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. For each command we briefly define the **"Purpose"** so you can quickly get an idea of what it is used for. We then show the **"Syntax",** which shows the proper usage of the command along with its options if applicable. The following conventions are used in the **"Syntax"** section:

[...] The parameters in the 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 TELECOM>EXPRESS> 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.

+, -, / The 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 command which seems logical, to remain compatible with older SpartaDOS versions, or to try and maintain command similarities for people who use 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. 104 KB (13 banks) of the SpartaDOS X cartridge is devoted to these external commands.

"Related" commands are shown 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 on this command has been made available. Since there are several versions of SpartaDOS X - old and new ones – it helps to distinguish them.

"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, which is important if you wish to be a SpartaDOS X user.

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.

We provide convenient space for your notes by starting the text for a new command, new section, etc. always on a new page.

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 adding temporarily a directory to the $PATH without manually rewriting all the paths that $PATH contains.

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
SpartaDOS X brings a full featured ARC utility to Atari 8-bit computers. ARC is based on and compatible with ARC.EXE by System Enhancement Associates which was written for the IBM PC. It is also compatible with ARC versions running on the Atari ST and other computers. ARC will take a group of files and quickly combine and 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.

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.

"Command" is one of the following:

A Add files) to the archive. Add all files from the file list to the archive.

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 if needed.

P Print files) 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; for example,

```
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.

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.

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. If you forget your password, you will not have a useful archive. For example,

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

In the preceding example the three files in the file list would be added 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 which means duplicate file names are not allowed (one will replace the other).

ARC is very useful for saving time while uploading/ downloading files with a MODEM and saving space for archival storage. ARC uses four stowage methods and automatically determines the best method(s) suited to each file. Our SpartaDOS X version of ARC is also fully compatible with ALFcrunched 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 four 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. ALFcrunch exclusively uses a variant of this method.

NOTICE: The name ARC, compatibility, and all other similarities to the ARC.EXE program by SEA (for MSDOS computers) are intentional. This trademark and the "look and feel" of the program have been licensed for SpartaDOS X by ICD, Inc. from SEA (System Enhancement Associates).

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, refer to the "Technical Information" chapter.

BASIC Command

Purpose
This command enters the ***internal*** BASIC in your XL or XE computer (1200XL does not have internal BASIC).

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

Type
External - uses CAR.COM on device CAR:

Related
CAR, SET

Remarks
If no filename is given, control is given to the ***internal*** BASIC of your XL/ XE computer. If you do specify a filename, the internal BASIC is enabled and the binary file you specified is 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. To automatically load and run a BASIC program from the command processor, read the I/O Redirection Section in the "Advanced Features" chapter.

The BASIC command can execute the internal BASIC even if the computer has 1088k RAM (of course depending on your special hardware configuration).

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, then the memory-save feature is disabled. If this feature is disabled, BASIC will be entered cold (there will be no program in user memory).

There is no default value of the $BASIC variable, unless RAMDISK.SYS is installed. If the variable $BASIC has not been set by the user (SET command in CONFIG.SYS) while RAMDISK.SYS is being installed, the ramdisk driver sets the variable so that it points to a BAS.SAV file residing in the ramdisk. You of course may change this with the SET command, 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
```

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

```
SET BASIC
```

(See the SET command for a further explanation.)

With the memory-save feature enabled, if a problem is encountered when loading or saving the memory file (BASIC.SAV by default), an error message will appear. 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 detail of the two situations that can occur follows:

- 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 disk fills up 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

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.

BOOT Command

Purpose
This command tells a SpartaDOS formatted disk to boot a particular program at start up (normally 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 SpartaDOS on the diskette it formats, so, if you want to create a bootable SpartaDOS diskette, you must COPY SpartaDOS to the diskette and use the BOOT command. If you just use SpartaDOS X, this will never be necessary (since SpartaDOS X boots from cartridge), but if you have SpartaDOS 3.2 (or 2.3), you may wish to create bootable diskettes. Of course you may still use the XINIT command to format and install DOS on your diskette.

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 (e.g. X32D.DOS) to boot!

- The file you set to boot may reside anywhere on the diskette - 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. Since you now have SpartaDOS X, we *strongly recommend* that you abandon SpartaDOS 1.1.

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 [/N] [d:] [path] [fname] [parameters]

Type
External - on device CAR:

Related
BASIC, COLD, SET

Remarks
If no filename is given, then control is given to the cartridge plugged into the SpartaDOS X cartridge. If you do specify a filename, then that binary file is loaded and run with the cartridge enabled. This is useful 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.

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 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, then the memory-save feature is disabled. If this feature is disabled, then the cartridge will be entered cold (there will be no program in user memory).

There is no default value of the $CAR. You may change this with the SET command (see the BASIC and SET commands for details on this).

If a problem is encountered when loading or saving the memory file, you will be told the problem and asked if you wish to cancel (i.e. go back to where you came from). Press the ESC key if you wish to cancel. The two situations that can occur are as follows

- 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 the cartridge (using the DOS command), the disk fills up 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 data in memory, or to go back to the cartridge (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 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.)

Whenever SpartaDOS is re-initialized (i.e. RESET), the default directory on *every* drive is reset to the MAIN (root) directory. The default directory of a drive is also 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 names, 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 diskette).

Syntax
CHKDSK [d:] [/XV]

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.

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 Multi I/O still works and is totally independent - it is a far more secure write-lock.)

The following is a sample output of the CHKDSK command using the /X option:

```
          Volume name: BOOT     A0 8E
      Filesystem type: SpartaDOS
   Filesystem version: 2.1

  Physical media type: fixed disk
      Number of sides: 1
     Number of tracks: 1
 Physical sector size: 512 bytes
  Sectors per cluster: 1
    Data cluster size: 512 bytes
       Total capacity: 65535 clusters
 Remaining free space: 44739 clusters

       Root directory: $0012
   First free cluster: $5037
  Ditto for directory: $06F4
     VTOC global size: 16 clusters
   VTOC first cluster: $0002

 Formatting date/time: 21-10-08 20:35
```

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 random number generated when the disk was formatted. The second is a sequence number which is incremented each time a file on the disk is opened for write.

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.

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 diskettes. The volume name may contain any ATASCII characters including spaces and inverse characters.

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

Type
Internal

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

COLD Command

Purpose
This command reboots the system (by doing a jump through $E477).

Syntax
COLD [/CN]

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, they are:

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.

Once SpartaDOS X has been disabled, it will be necessary to turn the computer off and back on to re-enable SpartaDOS X.

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 of the 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 (3-4Kbytes) 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. Then, 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
Compare the given files

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

Type
External - on device CAR:

Availability
As of SpartaDOS X 4.40.

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

CON (Console Modes)

Purpose
CON: drivers control.

Syntax
CON 40|60|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 and CON80.SYS drivers (see Chapter 8). The commands 'CON 64 ON' and 'CON 80 ON' 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 ON' 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.

Note: Users running QUICKED should be aware of the right order in CONFIG.SYS, where DEVICE QUICKED must precede DEVICE CON80.SYS to get both working properly.

COPY Command

Purpose
Copies one or more files to another drive and, optionally, gives the copy a different name if specified

COPY also copies files to the same disk. In this case, you must give the copies different names unless different directories are specified; otherwise, the copy is not permitted. Concatenation (combining of files) can be performed during the copy process with the "/A" parameter.

You can also use the COPY command 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 with a single disk drive and no ramdisk, see the MENU command. MENU has a provision for disk swapping, while COPY does not.

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

Type
Internal

Related
MENU, TYPE

Remarks
"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, which program to call. For example:

```
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.

The following details apply to the default "CAR:COPY.COM" program.

The first filespec specified is the source file name. If none is given, a default filespec of "*.*" is assumed (which will copy all files in that directory). The device for the source file should be given. However, it is possible to omit the source device if you use commas (instead of spaces) to separate parameters, for example:

```
COPY,,C:
```

will copy all files from the default drive/directory to the current directory of drive 3. The second filespec is the destination - if no filename is specified, a default filespec of "*.*" is assumed (which will copy the files without changing the names).

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

You may use wildcards ('*' and '?') 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 in the RENAME command is used. 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.

The meaning of the switches is as follows:

- /D - do not preserve date and time
- /I - ask before overwriting a file
- /M - delete the source file (move)
- /N - skip existing destination files
- /Q - do not print anything (except error messages)
- /R - dig recursively into subdirectories
- /V - summary (number of files and directories copied)

The "/R" option may need some explanations. The COPY command normally ignores directories - it only can copy files from one existing directory to another, but it cannot copy directories themselves. If you specify "+S" before the source file mask, then directories encountered will be copied as regular files, each one 23 bytes long.

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 all its contents, type:

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

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"). DELTREE, fortunately, can delete this.

The "/M" switch "moves" files. If the source and destination are on different disks, normal copying takes place and then the source file is deleted.

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.

If you are copying from a device other than "DSK:" (alias "Dn:" or just "n:"), then just one file is copied and the destination filespec is the name that the file will be saved under. For example:

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

is illegal because you may not have wildcard characters in a destination filename when COPYing from a character device (or for that matter SAVEing any file). However, if copying from one character device to another character device, filenames are not used. (Character devices never use filenames.) For example:

```
COPY CON: PRN:
```

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

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

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

will display the contents of "README.DOC" to 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 as follows:

```
TYPE README.DOC

TYPE README.DOC >>PRN:
```

with the second command sending the contents of the file to the printer.

You may also append files using the COPY command 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.)

If you only have one drive and wish to COPY files from one diskette to another, you must either COPY the file from the source diskette to a ramdisk and then from the ramdisk to the destination diskette, or use the MENU program as it allows disk swapping during copying.

The following command:

```
D1: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, the switches are treated as follows:

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

DATE Command

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

Syntax
DATE

Type
Internal

Related
CHTD, TD, TIME

Remarks
Calling the "DATE" command displays time and date in the European (dd-mm-yy) or American format (mm-dd-yy) depending on user selection. The default format is European. Use the environment variable $DAYTIME to change the format:

```
SET DAYTIME=1  American format

SET DAYTIME=2  European format
```

This command "DATE" produces the following output

```
Current date is 29-6-08
Enter new date (DD-MM-YY):
```

You may enter the new date or press RETURN if you don't want to set a new date. Enter the date format shown in brackets on your screen where "mm" is the month, "dd" is the day, and "yy" is the year.

This command will produce meaningless results if you do not have a clock driver installed in your system. The three clock drivers are "RTIME8.SYS", "ARCLOCK.SYS", and "JIFFY.SYS" - the first using the R-Time 8, the second using the ATARI Realtime Clock, and the third using the jiffy counter to keep its time. By default, one of these drivers will be installed when you boot, but this can be overridden by creating a custom "CONFIG.SYS" file and not including these drivers in the configuration.

DELTREE Command

Purpose
Delete subdirectory trees recursively.

Syntax
DELTREE [d:][path]dirname [/Y]

Type
External - on device CAR:

Related
RMDIR

Availability
As of SpartaDOS X 4.40.

Remarks
The command when executed asks for confirmation. If permission is granted, it removes the given subdirectory recursively with all the content, reporting progress successively.

The additional switch [/Y] suppresses the question, which the program normally asks before making deletions. Use it only when you are sure what you are doing.

Displaying "Can’t delete directory" happens, when an invalid directory entry has been found on the disk – 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 "CleanUp" program from SpartaDOS Toolkit to verify the structure of the file system and fix it. See more in Appendix C, "Using CleanUp with SpartaDOS X".

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 produces a list of all active drives, displaying the following information for every single drive:

- Drive letter.
- Total number of sectors.
- Number of free sectors.
- Number of free kilobytes.
- Percentage of the disk space already used.
- Volume name.

An overall summary is displayed at the end.

Adding the "/A" switch causes the program to list all the disks from A: to O:, displaying the appropriate error message for unreadable ones. Without "/A" the program lists only these disks, which allowed to read the information and no error occurred - 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] [/PC]
DIRS [+A|H|P|S] [-A|H|P|S] [d:][path][fname][.ext] [/PC]

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. With SpartaDOS 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. Example:

```
Volume:    TEST
Directory: MAIN

BACKUP TAR 6988k 10-09-06 15:55
  51483 FREE SECTORS
```

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.

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. (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 AtariDOS 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 attributes (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 as follows:

A Archived file. This attribute is cleared (-) whenever a file is created or updated. It is set when the file is backed by a program such as FlashBack!

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).

P Protected file. You may not ERASE or update protected files. Use the ATR command to protect or unprotect files.

S Subdirectory. This attribute is unchangeable.

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

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

Note: you must follow a subdirectory with a ">" or "\" character if you wish to see the contents of that directory.

DPOKE Command

Purpose
Puts a two-byte value (a word) into the given address.

Syntax
DPOKE [$]location [$]value

Type
Internal

Related
POKE, PEEK

Availability
As of SpartaDOS X 4.40.

Remarks
DPOKE allows you to change memory locations from the command processor, which can be useful in batch files and other applications. It is very easy to crash the system with this one if you do not understand what you are doing. Using DPOKE needs to preserve the common 6502 byte order (low/high).

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 (semigraphics, inverse video characters etc.) to be replaced with dots. This allows to print the DUMP output on a printer especially, if the printer interface does not allow full code translation or if it's not using a graphics mode to print.

DUMP is useful to quickly examine the contents of a file. To modify a file or examine and modify disk sectors, use "DiskRx" from SpartaDOS 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
```

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, if they are not too big (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. To change that, you have to declare an environment variable $ED, and give it a numeric value equal to the required number of lines, for example:

```
SET ED=20
```

Values from 1 to 22 are allowed. Values exceeding this range will cause the ED to assume the maximum possible size (i.e. 22 lines).

When there is a filename given on the command line, the program will attempt to load it. There are the following editing commands available:

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 up the low margin of the editor window.
Ctrl/D	Down – move down the low margin of the editor window.
Ctrl/V	Visible – make EOL characters visible.
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.
Shift/Insert	Insert the current line before the tagged one (see Ctrl/T), the cursor moves to the next line.
Shift/Ctrl/E	Erase – clear the editing buffer.

Note: Currently ED.COM is working in GRAPHICS 0 only.

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 no warning is usually given. If you do enter *.* as the specifier, SpartaDOS X will prompt you with:

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

With SpartaDOS 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 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.

Syntax
FIND [d:]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. FIND works like WHEREIS.COM (from SpartaDOS Toolkit) with a few exceptions. FIND's search is slightly different and it does not have a parameter to display file size, time, and date, for each file found.

The filename may include wildcards as desired. All filename matches found will be displayed with the full path from the root directory to the filename match. The number of matches found will be displayed at the end of the search. All drives will be searched unless a drive number is specified. FIND will also find and display hidden filenames.

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. It reads the input line by line, and applies the following to each one:

- Spaces at the beginning of the line are removed.
- When the "/S" switch is specified, all remaining spaces will be collapsed: multiple spaces are turned into one single space.
- If the resulting line is longer than "ncol" characters, it will be folded.
- When the "/J" switch was specified, 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 wraping 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, which allows you to initialize just about any type of disk 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 like AtariWriter or 850 Express!

The formatter has been enhanced to support larger file systems (up to 32 MB per partition is now possible), to handle greater sectors (up to 512 bytes), and to be able to use more drives. The drives are selected by the U function (as in Unit) using letters from A (drive 1) to O (drive 15). The density selection offers, apart from the three old ones, i.e. Single, Dual, Double, the fourth density named DD 512. This is the double density at 512 bytes per sector (just as the Dual is in fact double with 128 bytes per sector). The DD 512 density is usable with certain types of hard drive interfaces (KMK/JŻ/IDEa), and with TOMS floppy drives, either the original ones (TOMS 710, TOMS 720), or third party drives with TOMS Turbo Drive or TOMS Multi Drive extensions installed.

When you format a floppy diskette, you are first writing the sector structure to the diskette so the DOS has a place to put the program information. Next the directory structure is written to the diskette which is where the DOS keeps track of sector usage. You can also initialize a ramdisk or hard disk in 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 floppy disk 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 4.42 attempts to read the existing volume name and present 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 low-level (physical) formatted 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 free sector more, than you usually have on a freshly formatted diskette.

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. 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 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 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. The correct skew is required on most drives for the fastest possible reading and writing. 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** or **Atari**. Sparta is for SpartaDOS disk directory structure and Atari is for all the AtariDOS 2.0 clones and their directory structures in single and double density only.

Note: FORMAT does not write a "DOS" file to the diskette. If you want to create a bootable SpartaDOS disk, you must copy a "DOS" file from the SpartaDOS Construction Set to your formatted disk and then use the BOOT command. (See the BOOT command.) If you want to create a bootable AtariDOS diskette, you will need to boot AtariDOS and write the DOS.SYS and DUP SYS to the drive through its menu. SpartaDOS X will boot with any or no disk.

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

D **Density** may be one of four 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, or 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.) 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 OS enhancement), 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.

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 with 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 to determine whether the Tracks selection is wrong for the drive. It is important to enter the correct information or the disk 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 parameters required. It also writes the directory structure selected in Mode after the physical format and verify is completed. (The physical format and verify are functions of the floppy disk 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 disks. 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 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 incongruence. 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 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 turbo systems, which load the fast serial I/O patch onto the stack, will not work in turbo mode. Problems will occur with Top Drive 1050, TOMS Turbo Drive or any other using the same method. Such a drive can be used with 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 2000 Super, Indus GT, Happy 1050, Speedy, US Doubler – will work normally.

FSTRUCT Command

Purpose
Analyze a binary file.

Syntax
FSTRUCT [d:][path]filename.ext

Type
External - on device CAR:

Remarks
The command displays the information about the structure of the specified binary file (type, load address, length of the segments etc.).

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.

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. Initially, LESS was the MAN's internal code, but since it can be useful to view any text file (and not only the documentation), it has been separated from that program.

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 contents 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 width of the screen.

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

If the text, however, is longer than 23 lines, it will be "paged", and the viewer will not automatically terminate. There are the following keystrokes available to navigate through the viewed file:

- Down arrow or Return: scroll one line down (towards the end of the text)
- 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

- '<': 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, as the final receiver of the data stream sent through a pipe, for example:

```
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).

LOAD Command

Purpose
Loads a file (does not 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 continuous 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 keep 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 [command|/?] [/P]

Type
External – on device CAR:

Availability
As of SpartaDOS X 4.40.

Remarks
MAN.COM is a simple text viewer. Its main purpose is to view documentation files to programs and commands given as the parameter. The program's operation is similar to the man command known in UNIX (where it is an abbreviation for manual), thus the name.

To use it, it is required to define the environment variable $MANPATH first. It will contain 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 executables:

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

Now if you put to any of these directories a file with *.MAN, *.DOC or *.TXT extension, and then give its name (but omitting the extension) to MAN.COM, the file will get searched for and displayed when found.

For example, the HDSC.ARC archive contains a text file named HDSC.DOC filled with informations about the program. When you have 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. Now, when you want to read the instructions, you do not need to remember, where the original archive is, unpack it again etc. It is enough to execute this command instead:

```
MAN HDSC
```

and the 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 width of the screen.

Invoking the command without parameters will produce a list of all available manuals.

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 contents of the file to the screen, without any processing, and when the file ends, it quits to the 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, to printer or whatever. For example, to make a hard copy of the file HELP.DOC, you just need to do the following:

```
D1:MAN HELP >>PRN:
```

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

When paging is required, apart from normal methods, such as

```
D1:MAN HELP | MORE
```

an external viewer can be registered for the MAN.COM. Currently there are three available ones: 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, it is enough to do that:

```
D1:SET PAGER=MORE
```

or

```
D1: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, but we supply the method of registering it, because it can serve as an example on how to register an external viewer, which needs additional parameters.

Launching an external viewer works so that the MAN.COM must first construct a command line out of the template stored in $PAGER, and the specification of the file it found among the help files. But, first of all, since the /P switch must be given to TYPE *at the end* of the command line, there must be a method of telling the MAN.COM where it has to insert the filespec. 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:

```
D1: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:

```
D1:SET PAGER=LESS;/C;%
```

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

MAP Command

Purpose
SIO.SYS control.

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

Type
External – on device CAR:

Availability
As of SpartaDOS X 4.40.

Remarks
The command "maps" the given drive identifier (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 floppy drive, which has been masked out by a parallel (PBI) drive having the same number. For instance, the command "MAP 1 SIO D2:" creates a logical D2: drive, which uses physical disk drive number 1 (the D1: can be a partition of a parallel hard drive).

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.

The SIO parameters to be changed can be stored in a file, and the name of that file can be given to the MAP command as the first (and only) argument. That should be a text file, and each line should 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 "/C" option!

MDUMP Command

Purpose
Display memory in hex and ATASCII.

Syntax
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 contents quickly.

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. E.g. the mode it is being used by DOS (NONE, OSRAM, BANKED) with more details using the [/X] option, the bottom limit of available user and extended memory. 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.

In the following example,

```
Main: $0F6D,$1456
 Ext: $6123,$6455
 Use: BANKED

7 banks available
```

the installed drivers use memory from $700-$F6C and $4000-$6122, and the held-in-memory applications used reside in memory from $F6D-$1455 and $6123-$6454.

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

An example of MEM /X (different config than above example):

```
Main: $1099,$1099,$0000
 Ext: $75E2,$75E2,$0000
 Use: BANKED, bank $2F
 Top: $91CF ($BC1F),$7FFF
Free: 35718 (43910),2589

12 banks available (196608 bytes)
```

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. "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.

Note: If a permanently installed driver is installed after LOADing a held-in-memory application, both low memory values will be raised above it and any held-in-memory applications will become permanent.

Although there are two possible extended memory regions, only one may be used by SpartaDOS X. 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).

Due to a lot efforts 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. See preface for it.

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

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 area logged. 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 menus include: COPY, DELETE, EXEC, FILESPEC, LOG, PRINT, RENAME, SHELL CMD, TAG, UNTAG, and VIEW.

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

Xtra command menus include: 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, or would not use 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".)

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 the same as 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 drive number logged 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 contents 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".

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; instead use Log for that. 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 drive number logged 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 relog 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, 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 directories currently logged.

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

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 created. 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

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 normal ways in which files are. To rename a subdirectory you must copy all files from inside it to another area, then delete all the files in it, use RMDIR to delete the directory, then create a new name with CREDIR, and copy the files back to it. Otherwise, you can use RENDIR.COM.

```
MD TEST
MKDIR 3:>MODEM>TEST
```

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 is in the MAIN directory.

MNT Command

Purpose
To address the KMK/JŻ IDE or IDEa devices if connected.

Syntax
n.a.

Type
n.a.

Remarks
The description of this program belongs to the documentation of the respective hardware and thus will not be discussed here.

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: INVERSE, LESS, FMT.

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. Then 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 (such as X, CAR, MENU, DUMP, CHTD, etc.) that you may need. 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
```

Where 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

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.

Note: It is very dangerous to change diskettes (during a PAUSE) on the drive from which the batch file was running. If using PAUSE to change diskettes, run the batch file from a ramdisk or another drive which will not be changing.

PEEK Command

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

Syntax
PEEK [$]address or PEEK symbol

Type
Internal

Related
POKE, DPOKE

Remarks
PEEK allows 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.

It is a good idea to PEEK a location before POKEing it if you are not sure what you are doing. You can usually recover by POKEing the old value back in (unless the computer has crashed).

The PEEK symbol command displays the address and memory index of the given symbol, in identical manner as it was done by the FSYMBOL command. The FSYMBOL command has been removed.

POKE Command

Purpose
To change the contents of a memory location.

Syntax
POKE [$]location [$]value

Type
Internal

Related
DPOKE, PEEK

Remarks
POKE allows you to change memory locations from the command processor which can be useful in batch files and other applications. It is very easy to crash the system with this one if you do not understand what you are doing. A few examples of POKE locations and useful values follow:

POKE 77 (attract) 0=attract mode off for a few minutes

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=weaker off (XL/XE only)

POKE 752 (crsinh) 1=cursor off, 0=cursor on

POKE 702 (shflok) 0=lower case, 64=upper case

POKE 65 (soundr) 0=SIO sound off, 1=SIO sound on

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

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 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 'I')
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)

Note: "P" causes the current drive to be read every time RETURN is pressed to detect a disk change. Modern hard drives do not care about that. But older hard drives, which need to be parked before the system shuts down, should be taken out of the path before parking them, since the "P" will read the disk and unpark the drive. A batch file changing the prompt, then parking the drive, would be very useful for this purpose.

If no parameter is specified, then the current prompt string is 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 really just a convenient form of the SET command, for example the above command could be performed by:

```
SET PROMPT=$L$P\
```

just as easily. The default value of the "prompt" variable is "D$N:" which displays the same prompt as older SpartaDOS versions. 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
```

You can not use non-inverse lower case letters in the prompt because the command processor automatically converts all lower case characters to upper case before processing. Inverse and cursor control keys (preceded by the escape key) may be used in the prompt.

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 unrecognized 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.

RDDUMP and RDLOAD Commands

Purpose
Saving and restoring the ramdisk contents.

Syntax
RDDUMP d: [d:][path]fname[.ext]
RDLOAD [d:][path]fname[.ext] [d:]
Type
External - on device CAR:

Related
RAMDISK.SYS

Availability
As of SpartaDOS X 4.42.

Remarks
These two commands allow the user to save the ramdisk content into one file on a real storage device before powering off, and restore it with the next system start to use it again. This is helpful when using storage devices that are much slower than ramdisks. At system startup the dumped contents will be reloaded into memory; the ramdisk works as a kind of volatile hard drive.

Configure the system to automatically install a ramdisk during bootup. Setup directories and copy files to it as appropriate (e.g. your preferred tools & utilities). When finished dump the ramdisk to a file. See the example:

```
D1:RDDUMP O: C:>SYSFILES>RAMDUMP.BIN
```

Use the RDLOAD command to restore the contents of the ramdisk from this file. Omit the second parameter to get it automatically from the environment variable $RAMDISK (which is set by RAMDISK.SYS during boot time). Example:

```
D1:RDLOAD C:>SYSFILES>RAMDUMP.BIN
```

This operation requires two conditions to be met:

The destination drive must be identified as ramdisk (not a floppy disk or hard disk),

and

its capacity has to match the size of the dump file.

To minimize the time required to restore the ramdisk from a slow storage device, RDLOAD only reads sectors, which at the time of creating RDDUMP were marked as occupied. This is worthwhile when dumping a ramdisk filled to ca. 50%. To restore a nearly filled one consumes eventually more time than a simple sector copy.

To avoid restoring the ramdisk, when it was not reformatted after the last restart, edit the AUTOEXEC.BAT file and put the following lines into it:

```
IF RAMDISK
  IF NOT EXISTS $RAMDISK>*.*
    RDLOAD C:>SYSFILES>RAMDUMP.BIN
  FI
FI
```

Of course, you can specify the name of a characteristic file (or directory - in this case +S is required after EXISTS) instead of *.*.

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. If you try to save a file with a name already existing error 151 "File exists" will come up and nothing is saved. 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 the same to directories, what RENAME does to files. RENDIR.COM origins from the SpartaDOS Toolkit and is now present on the CAR: device.

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
The directory must be empty before it 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 even though subdirectories are supported by that driver.

```
RD TEST
DELDIR 3:>MODEM>TEST
```

The first example removes the subdirectory called "TEST" on the default drive. An error 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 under 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 the 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. It is possible that some sectors have been allocated for this file. These should be de-allocated in the bit map. See more in Appendix C.

RPM Command

Purpose
To check the RPM of a floppy drive.

Syntax
RPM [d:]

Type
External - on device CAR:

Remarks
This command will continuously check and display the number of revolutions per minute (RPM) made by a drive until any key is pressed. This is mainly useful as a diagnostic tool to determine if a floppy drive is operating at the proper speed (288 RPM for most Atari drives, including the 810 and 1050, and 300 RPM for the XF551). This command will also give accurate information for a hard drive. Using RPM on a Multi I/O ramdisk will simply provide the relative access speed of the ramdisk, a worthless but interesting piece of information. RPM will not work on internal ramdisks. It will not work properly on enhanced drives while track buffering is enabled, either.

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 and an error occurs, 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.

RUN Command

Purpose
Run the code at the specified address.

Syntax
RUN [$]address

Type
External - on device CAR:

Availability
As of SpartaDOS X 4.40.

Remarks
The command makes a JMP to the specified address. It is not sanity-checked before, it is assumed, that the user is invoking the command on purpose and is sure what he is up to.

S2I Command

Purpose
To address the SIO2IDE device if connected.

Syntax
n.a.

Type
n.a.

Remarks
Supported are SIO2IDE with firmware 3.x and 4.x. The description of this program belongs to the documentation of the respective hardware and thus will not be discussed here.

SAVE Command

Purpose
This command saves binary data from memory to disk.

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

Type
Internal

Related
LOAD

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.

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 for one program to communicate to another with. For example the $CAR variable tells the CAR command where the memory-save file is. There are three forms of this command. For example the command:

```
SET
```

displays the contents of all environment variables, and:

```
SET CAR=A:CAR.SAV
```

sets the variable $CAR to the value "A:CAR.SAV".

The command:

```
SET CAR
```

deletes the environment variable $CAR from the system. (This will cause the CAR command to not use a memory-save file.)

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 exist, upon completion of the AUTOEXEC.BAT file the specified directories will be current on specified disks.

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]]]

Type
External - on device CAR:

Availability
As of SpartaDOS X 4.40.

Remarks
The SIOSET purpose is 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 runtime) you may want to change them by hand.

With no arguments given, SIOSET displays the current configuration for all drives. 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 baudrate.

XF The drive uses the XF551 protocol.

US The drive uses the UltraSpeed protocol.

INDUS The drive uses the Indus protocol.

When UltraSpeed is used, the additional *usindex* parameter allows to determine the serial speed. For example, the so called 3xSIO mode (3x19200, i.e. 57,6 kbps) requires $08 as *usindex* value.

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

SL Command

Purpose
This command generates a list of SpartaDOS X symbols.

Syntax
SL

Type
External – on device CAR:

Availability
As of SpartaDOS X 4.40.

Remarks
The command generates the list of SpartaDOS X symbols.

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:

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 – in descending order

The file specification may be omitted, the current directory is sorted then, but a 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.COM origins from the SpartaDOS Toolkit and as of SpartaDOS 4.40 is now present on the CAR: device.

SWAP (Swap Drives) Command

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

Syntax
SWAP [d,d]

Type
Internal

Remarks
SWAP, without any parameters, will display the drive map list showing drives 1 through 9. The default is 1=1, 2=2, etc.

For example, to interchange drives 1 and 9, type the following command:

```
SWAP 1,9      (or)
SWAP A,I
```

The order is not important, so 1,9 is equivalent to 9,1.

The drives will stay mapped that way unless 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 Multi I/O drive remapping, so take that into consideration when using the Multi I/O. It is very easy to lose track of which floppy, ramdisk, or hard drive partition is at which logical drive.

Note: The SWAP command has no effect on drives from J: to O:.

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 or RTIME8.SYS driver installed before you can use this command. It calls one of these drivers directly (through the I_GETTD symbol) and will not load without one of these drivers. (These drivers are loaded by default unless you are using your own CONFIG.SYS file.)

The Y2K problem is fixed with SpartaDOS X 4.4x. The format in which the time is displayed depends on the value of the $DAYTIME variable (see DATE command).
Additionally, the "X" letter reflects the Caps/Inverse state. See also Z.SYS (chapter 8).

Note: 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

Type
Internal

Related
CHTD, DATE, TD

Remarks
This command produces the following output

```
Current time is 14:34:30
Enter new time (HH:MM:SS):
```

You may enter the new time or press RETURN if you don't want to change it. Enter the time in the format "hh:mm:ss" where "hh" is the hour (24 hour clock), "mm" is the minutes, and "ss" is the seconds. (SpartaDOS 3.2 uses AM/PM - SpartaDOS X uses a 24 hour clock.)

This command will produce meaningless results if you do not have a clock driver installed in your system. The clock drivers are "ARCLOCK.SYS ", "RTIME8.SYS" and "JIFFY.SYS" - the first being for the ATARI Realtime Clock, the second being for the R-Time 8 and the third uses the jiffy counter to keep its time. By default, two of these drivers will be installed when you boot, but this can be overridden by creating a custom "CONFIG.SYS" file and not including these drivers in the configuration. "ARCLOCK.SYS" will not be loaded automatically but only via custom "CONFIG.SYS".

As of SpartaDOS X 4.42 the command verifies the numbers entered by the user, so values like 99:99:99 will not be accepted.

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 were not seen by UNERASE, they were not recoverable for one of two reasons:

1) The file's directory entry has been allocated to another file which was copied to the directory after the original file was ERASEd.

2) A sector of the file has been allocated to another file since the original file was ERASEd.

Note: UNERASE.COM in SpartaDOS X 4.2x contains a long-known bug, that causes it to screw up the disk’s bitmap, when the file being undeleted resides in a series of data sectors, and the group of bits representing these sectors happens to cross a sector boundary in the bitmap. This bug is fixed, 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.

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. This only applies to floppy drives. Because of the extra time required to perform the verification, the system runs much slower when programs write data to disk. The default is OFF - this command is typically used when you are having floppy drive problems.

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 a command can run under. They are:

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 search path is not used - you must specify the exact location of the file if it is not in the current directory of the current drive.

- 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 can not 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.

X.COM remains resident in memory while the program called is running, so MEMLO will be slightly higher until the program is exited to the command processor.

Executing a cold start (a jump to $E477) while using X.COM will disable the SpartaDOS X cartridge and the external cartridge, if present.

XFCONF Command

Purpose
To select density in floppy drives.

Syntax
XFCONF [d:] [/12345]

Type
External - on device CAR:

Availability
As of SpartaDOS X 4.40.

Remarks
Some floppy disk drives have problems when it comes to automatic density detection. This problem particularly beats the Atari XF551 disk drive (thus the name of the command), but the program can be handy in case of any other drive too, when it happens to be somehow stuck in an improper density. Then, the XFCONF allows to force the drive manually into the proper one.

The command invoked without arguments displays menus, where you can choose drive number (1-4) and the desired density. After the new configuration was sent out to the drive, it is checked, whether the drive has really selected it (some drives do accept and positively acknowledge densities impossible for them, for example 360k for an Atari 1050 Top Drive, or 720k for an Atari XF551 – the drive's controller selects something closest to the requested configuration and replies "OK" to the computer – which is surely far from being OK). The message *Drive cannot do this density* means, that the requested density was accepted, but the drive is in fact unable to realize it.

Other messages:

Drive not configurable	The drive is a stock Atari 810 or Atari 1050, no modifications. Density changes are not possible.
Drive is not a floppy disk	An attempt to change the density of a ramdisk or hard disk has been tried.

Drive rejected the density The drive explicitly denied to select the density.

XFCONF accepts command line arguments too. Giving a drive identifier alone causes the computer to check, if the drive is a floppy drive and if it is configurable. The reply *Drive is configurable* confirms that. The density change can be accomplished by adding a switch followed by a digit from 1 to 5, where the digits mean densities as follows:

/1 single density (SS/SD, 90k)

/2 dual density (SS/ED, 130k)

/3 double density (SS/DD, 180k)

/4 double sided double density, 40 tracks (DS/DD, 360k)

/5 double sided double density, 80 tracks (DS/DD, 720k)

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 over 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 which accepts two files on input and creates an output file containing the contents 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 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. The variable $BATCH, then, can be used to cause a batch file with a different name to be executed on boot by using 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, if 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
```

Save this as a batch file named for example 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 on the 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. It is therefore a 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 it is the "A"-key you want the batch file to wait for, the command doing that may be in any of the form below:

```
IF INKEY 65
IF INKEY $41
IF INKEY 'A'
```

This serves well in simple comparisons, but for more complex tasks, like for example a menu with many options, you should use the INKEY command described below.

Comparisons

Checking the value is accomplished with the double equation sign (==), which is 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 for MS-DOS compatibility.

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
```

That is the definition. Giving 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 its beginning. This means, that the farer within the batch file is the label, 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, and so you can this way make for example 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, and the environment space is only 256 bytes. To avoid filling it with unnecessary garbage in SpartaDOS X versions before 4.42, it is a good practice to delete all variables created in your batch file before exiting it; just as it is shown in the example above (the fifth line counting from the end).

As of SpartaDOS X 4.42: a) an extension ENV.SYS is available, which provides as much as 15 KB for environment variables, and b) the Command Processor, upon BAT file completion, automatically removes variables starting with '%', ':' and '_'. Two latter 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 a 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 we do not recommend to exceed some 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 taken 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

You may redirect the standard input and output of SpartaDOS X commands on the command line. 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 may 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
```

For example, if you named the above batch file MORE.BAT and wanted 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 MS-DOS 5.0. It can be seen as extension to the I/O redirections described above. 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, and you do not need to trouble yourself with a temporary file.

For example, you want 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:

```
D1: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, and require 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. For example, if you prefer TYPE over MORE for a reason, the command line will look like this:

```
D1:ARC | TYPE - /P
```

The pipe can connect up to nine commands. For example, if the INVERSE command displays the given text in inverse video, you can do this:

```
D1:ARC | INVERSE | TYPE - /P
```

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 typing

```
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 it can be accessed much more rapidly than the others. Order is important, since the entries will be searched one after the other. For example,

```
PATH CAR:;A:\DOS\;A:\TOOLKIT\;D9:>;A:>;:>
```

will search the CAR: device, the directory DOS on D1:, the directory TOOLKIT on D1:, the ramdisk at D9:, 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 example in the last line on page 5-5 to do this would produce

```
PATH CAR:;;A:\DOS\;A:\TOOLKIT\;D9:>;A:>;:>
```

where the two semicolons after "CAR:" mean 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, for example, this command:

```
D1: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:

```
D1: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 language manual for details.

In the 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 once 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.

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 additional attributes are available, described below.

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. This mode allows both 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
In SpartaDOS versions older than 4.42, when opening the formatted directory input with the BASIC's OPEN instruction, only two values of aux2 were allowed: 0 and 128. The first one meant, as it was already said before, the "short" directory listing, i.e. the type displayed by the DIRS command. The other produces the "long" listing, i.e. the type displayed by DIR.

These two values have the same meaning in SpartaDOS X 4.42. But in fact, when aux1 is equal to 6, the bits of aux2 have the following functions:

Bit	Value	Description
7	+128	long directory format
6	+64	display attributes (p, a, h)
5	+32	make a space between filename and extension
4	+16	place a dot instead of that space (only when +32)
3	+8	long directory: suppress seconds short directory: directory extensions, ':' if +2
2	+4	24-hour time format
1	+2	two spaces before filename, '*' for protected files
0	+1	long directory: full length of the file (10 digits) short directory: display the file's length (in sectors)

Bit 2 (+4) when set, selects the 24-hour clock when the US time format (AM/PM) is selected globally, i.e. when the $DAYTIME is 1. When the global time format is the EU format, this bit is ignored.

The operation of bits 0 and 3 depends on the state of the bit 7. When it is on, setting bit 3 to one suppresses displaying seconds of the file's time stamp. Setting bit 0 causes the file's length to be displayed in bytes as a 10-digit number - otherwise the values exceeding 999999 bytes are 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 please remember, that selecting some other formatting attributes may make the line longer than 40 characters. The directory line may theoretically have up to 64 characters and this is the amount to be reserved 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
```

If you were an user of earlier SpartaDOS versions, you can notice, that the short directory format, the one compatible to ATARI DOS 2, has been slightly changed as of SpartaDOS X 4.41. Earlier versions never displayed directory extensions in this format, the text "DIR" in inverse video was used instead. In the current versions, the directory extensions are always displayed (either in long and short directory format), and a ':'-sign is put in front of the directory filename. This format is MyDOS-compatible.

The "old" format for short directory listing can be still invoked by setting 3 as aux2. But in this mode the directory extension are not displayed, even if they exist. So this feature is more fancy than useful, and therefore could be removed in future versions of the DOS.

Accessing the Raw Directory

Setting bit 4 of **aux1** 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 attributes desired. If a long directory is wanted in scan mode, then 128 should be added to **aux1** instead of to **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 IOCB selected should be closed for 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 two or more files on one disk the same name.

If you receive disks having this problem from other users:
It will then 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 unwanted 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 IOCB selected should be closed for 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 important 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 IOCB should be closed. 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 selected files previously protected with the ATR command or the Protect Files XIO command above. The file or files may now be erased, renamed, or changed. Wildcards may be used. The IOCB should be closed.

Set File Position – POINT

Purpose
To allow direct access to specific points within a disk file (or past the end of a file 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, which 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 disk. The file must be open for this operation.

Because of a limitation in Atari BASIC, BASIC XL, and BASIC XE, the first method shown, 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 greater location 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 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 a 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 and an Atari DOS disk may prove to be time consuming since, to determine the relative offset into the file, it is necessary to read the file from the beginning every time a POINT is used. This also causes segmented binary files to take much longer to load from Atari DOS disks 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, of course, no longer be valid.

Sparse Files
On a SpartaDOS diskette, 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 of these 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 unnecessary.

Get File Length

PurposeTo 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 when 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 AtariDOS 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 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, and the directory "SHEMP" will be created.

The IOCB should be closed for 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 when SpartaDOS X is not used.

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 IOCB should be closed for 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. Once the formatter is exited with the ESC key, control will be returned to the program. This allows disks of all types to be formatted from within any program. The IOCB should be closed for this operation. WARNING! Formatting a disk will destroy all existing data on the disk. Hiding or protecting a file will not save it from being destroyed during a disk format.

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 to 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.

A value of 128 indicates, just as before, 128 bytes per sector. Any other value is the high byte of the logical sector size value in bytes, minus 1. This allows to easily encode sector sizes from 128 bytes to 64 kilobytes, as follows (*=not supported):

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. It is followed by an assembly language listing of the code contained in the DATA statements and later in 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 will follow as an 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 identifier if none is given, and place the result at COMFNAM. Each call will advance BUFOFF to point to the next entry on the command line, so that each call to the crunch routine will get the next entry on the line. If there are no entries remaining, 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, when ran under SpartaDOS X 4.2x, were not able to control timestamps in the files copied.

SpartaDOS X 4.4x restores the function of these registers, but DATESET still has the highest priority, and when it is set, the TDOVER value is ignored.

The TDOVER is cleared after the program quits in SpartaDOS X 4.4x, but not in 3.2. Thus it is a good programming practice to clear this register always when the program is about to quit to DOS (of course, only 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 expansion memory banks free. This is the same number shown with 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.

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. It resets BUFOFF to 0 so that the name of the command is printed, too.

```
;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
; zero BUFOFF
     lda #0
     ldy #bufoff
     sta (comtab),y
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 follows:

```
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
```

These functions each contain a jump (JMP) followed by the address of the function. It is a good idea to always check for this JMP before 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, but they 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 4.2x).

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 known not from the Carry state after the call, but rather from the impossibility to do the call due to inability to find the symbol.

In the I_GETTD and I_SETTD procedures a set Carry means that the clock driver is busy at the moment. You should call the routine again. It is a 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 a reason (e.g. a hardware failure).

The I_FMTTD is loaded to the memory along with the 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 set, this indicates an error (the clock driver being in permanent busy state). If the Carry is cleared, there is still time and date information in the buffer, in form of an ASCII, EOL-terminated string.

Page Seven "Kernel" Values

Several page seven locations allow "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
sio_index	$70F	"swap" table (9 bytes) see below
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 subroutine jump (JSR) to address $703 with the desired command in the 6502 Y register and the desired device number in *device.* For example, with a $10 in *device,* a value of 100 in Y 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 and 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 wide each, while the other sectors in this density are 256-byte 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 error code otherwise.

The block_io function codes are here:

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 sio_index table, changed by the Command Processor's SWAP command and referenced to by SIO drivers, is still 9 bytes wide and cannot be enlarged, even though the number of valid SIO drive numbers has been increased to 15. This is the reason why the drives J: - O: cannot be "swapped".

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.

The low nibble of the *device* number is the unit number of the device, such as 2 for D2:. The high nibble is one of the following:

0 SIO block device (SPARTA.SYS, ATARIDOS.SYS, etc.)
1 Clock driver (RTIME8.SYS, ARCLOCK.SYS, JIFFY.SYS)
2 ROM cartridge driver
3 Console driver
4 Printer driver
5 RS232 driver (COM.SYS)
6 NUL: device
7 reserved

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. The drivers bundled with version 4.41 should be considered beta-versions, some functions described below may be missing in them.

The screen editor provided by the CON64.SYS or CON80.SYS 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 questions.

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 an 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 at the same time the function code of the XIO call that can enable and disable the extended console mode:

```
XIO nc,#0,12,m,"E:"
```

For the nc value you should substitute what XIO 33 returned as 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, which 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, which 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 ClearScreen 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 one line up. The following 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 moves then 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 Disk Format

There are four distinct types of sectors on a SpartaDOS format disk. 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. The boot sectors are always single density, for storage devices having densities 128 or 256 bytes per sector.

Sector 1 from offset $30 to offset $7F and all of sectors 2 and 3 are the boot code that loads a file under SpartaDOS 2.x and 3.x 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, 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 introduces sectors larger than 256 bytes per sector to the ATARI 8-bit world. 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 the 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. For all ATARI 8-bit DOS systems before SpartaDOS 4.4x the detection process was based on the 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.

Under the bottom line 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. At second more boot region space is available (512 instead of 384 bytes). Therefore creating 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.

These are the sector 1 values, given as offsets into the sector:

9 The sector number of the first sector map of the MAIN directory. 2 bytes.

11 The total number of sectors on the disk (2 bytes).

13 The number of free sectors on the disk (2 bytes).

15 The number of bit map sectors on the disk.

16 The sector number of the first bit map sector (2 bytes).

18 The 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).

20 The 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).

22 The disk volume name. SpartaDOS uses this as part of the disk change identification procedure (8 bytes).

30 The 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, for example) this location will contain a 1.

31 The size of the sectors on this disk (other than the boot sectors). A 0 indicates 256 bytes per sector, a 128 indicates 128 bytes per sector, and a 1 indicates 512 bytes per sector. Generally, everything else than $80 is the high byte of the sector size measured in bytes, minus 1.

32 The file system revision number of the disk format. SpartaDOS 1.1 disks will have a $11 here. Disks formatted for SpartaDOS 2.x, 3.x, and SpartaDOS X 4.2x will all have a $20 here (version 2.0), since they all use identical disk formats. SpartaDOS X 4.4x has $21 here (version 2.1).

33 As of file system version 2.1: the sector size as 2-byte-number in low/high format.

35 As of file system version 2.1: the number of sector entries per file map sector as 2-byte-number on low/high format.

37 As of file system version 2.1: number of physical sectors per logical sector (cluster). Note that only one value – $01 – is supported at the moment.

38 Volume sequence number. This number is incremented by SpartaDOS every time a file is opened for write on the disk. This is used to identify the disk.

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.

40 The 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 from the SpartaDOS Construction Set and the BOOT command.

Values 31 - 37 are read-only in SpartaDOS X 4.4x, it is not recommended to change them. Bytes 42-63 are reserved for future extensions. Therefore their values should not be altered for upward compatibility.

Caution: Locations 33-37 have different meaning in SpartaDOS 1.1 and later versions of the file system. Even though these bytes are not used, they retain values default for SpartaDOS 1.1. This is why the file system version number must be checked by the programmer before these locations are used in a program!

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 maps 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 under the POINT command for a description of sparse files.

Directory Structure

The directory structure of SpartaDOS 4.4x is identical to the older SpartaDOS 2.0 file system. The only difference is that the first entry (the header) of the main directory has a valid time stamp: it is the date and time, when the file system has been built on that disk. 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 SpartaDOS disk format is to use a sector editor and a test floppy to explore. "DiskRx", the SpartaDOS disk editor included in the SpartaDOS Toolkit, is an excellent sector editor tailored specifically for SpartaDOS disks. It will identify boot, bit map, sector map, directory, and data sectors. A good understanding of SpartaDOS disk structure and "DiskRx" can prove to be invaluable for recovering files from disks with bad sectors or damaged directories. Exploring disks can also be a lot of fun. Remember that "DiskRx" cannot handle 512 bytes per sector partitions.

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 in its stead it contains the third byte of the number of sectors per track. Otherwise, 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. To determine the current disk configuration, where you can judge on the size of the first sector, the usage of the READ PERCOM is strongly recommended.

The program below is an example of a subroutine, which 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 startup, 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 drive 1 when you boot. It must be on a SpartaDOS format diskette in the "MAIN" directory.

CONFIG.SYS File
The "CONFIG.SYS" file basically consists of commands. There are currently four commands. They are

```
USE OSRAM|BANKED|NONE
SET var=env_string
DEVICE driver
MERGE 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 your 130XE or memory expanded 800XL or 400/800 computer. NONE refers to low memory, above the normal low memory part of SpartaDOS and below program memory. USE NONE will probably be incompatible with many programs, since it uses up a great deal of main memory which

may be used by applications. Any line beginning with a semicolon (;) is considered a comment and ignored.

The default ROM CONFIG.SYS pseudo-file is

```
DEVICE SPARTA OSRAM
DEVICE SIO
DEVICE ATARIDOS
DEVICE INDUS
DEVICE JIFFY
DEVICE RAMDISK
```

The default RAM usage is as follows:

OSRAM	Stock XL/XE computer
BANKED	Banked memory expanded computer (RAMBO XL, AXLON, or compatible upgrades)
NONE	400/800 (48 KB) with no expanded memory

The default CONFIG.SYS file is to be found on the CAR: device (CAR:CONFIG.SYS). It will be read from there, when no user-defined CONFIG.SYS is found on the boot disk.

Note: 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 (explained later in this section).

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 does not have more than 64 KB of RAM, or the RAM extension is about 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, then, 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).

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 disk 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
Its use is optional, but *when it is used, it must be the last keyword in the current configuration file*, because it aborts its processing and merges

another one. This allows you to form a chain of text files to be processed at system startup, for example:

```
USE BANKED
MERGE DEFAULTS
```

The system will configure memory and then load a file named DEFAULTS.CFG. That file should contain actual configuration commands – and also may contain another MERGE at the end, if it is necessary to merge a portion from yet another file.

This feature is useful in conjunction with the multiple configuration files managed by the Config Selector: such configuration files, when they differ only at the beginning, may define differences in small number of commands as shown in the above example, and then merge the rest of the contents from common source. This allows to keep several configurations consistent by editing only a single text file.

The limitation to MERGE is 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. The file name extension (*.CFG) is optional.

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. The same memory area, however, 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 screen effects named above do appear.

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.

Config Selector
SpartaDOS X 4.4x offers a built-in config selector. This feature allows the user to store several alternative CONFIG.SYS files on the disk, 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.

At boot time SpartaDOS searches the main directory of the boot disk for a subdirectory named SPARTA.DOS. When it is found, and contains *.CFG files, a menu is displayed where each of the *.CFG files (up to nine) is assigned a number. Hitting the appropriate key chooses the corresponding file to be used to configure the DOS. If you press any other key (ESC, SPACE, RETURN) or wait few seconds, the DOS closes the menu and continues with standard procedures.

Important system variables
Some global system settings are controlled with environment variables (a list of them can be invoked into the screen by typing the SET command without the parameters, and hitting Return at the DOS prompt). 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 4.42 is below.

Note: Apart from these listed below, 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 user executes the command "DOS". Saved areas are: page 0, 4, 5, 6 and everything from LOMEM to APPMHI. 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, which 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 wiches to return to the DOS. Saved areas are: page 0, 4, 5, 6 and everything from MEMLO to MEMTOP. 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, or in other words 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, of course, should perform basic shell tasks like offering file management functions and program execution. The MENU can, as of SpartaDOS X 4.42, 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.

$DAYTIME

Defined by the user
Controls SPARTA.SYS, COMMAND.COM, TD.COM
Remarks Controls the date and time format. There are three values possible: "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). No variable has the same meaning as "0". The EU format is the default.

$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 with commas or semicolons - where the MAN command has to search 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 with RAMDISK.SYS. The default is "O:>"

$SYSERR

Defined by the kernel
Controls I/O library
Remarks Points to the text file, where the system error messages are fetched from. 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 consecutive errors, starting from error 128 in the 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 a good idea to put the SYSERR.MSG replacement on a really fast storage media, such as a ramdisk or really fast 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 a good idea to locate this directory on a fast storage media.

DRIVERS
SpartaDOS X 4.4x introduces a lot of all new features to the ATARI 8-bit world. Therefore quite some efforts have been done to change the 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. You cannot declare fewer than 3 buffers, and the 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 drive as described. Just increase your "nfiles" value by one or more. Increasing "nbufs" will speed up disk access for additional open files but is not required.

ATARIDOS.SYS Driver

Purpose
This driver contains the code to recognize Atari DOS 2 format diskettes. 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. 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 disks. ATARIDOS.SYS does not provide support for DOS 3, DOS XE, or OSS version 4 DOS.

Summary of formats supported for

- read and write: SS/SD (90 KB), SS/DD (180 KB)
- read only: SS/ED (130 KB)

Disks e.g. formatted using MYDOS must also stick to it. Otherwise it will not work even when a directory is shown correctly.

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 [/CA]

Type
External - on device CAR:

Remarks
You must include one of the SIO drivers in your "CONFIG.SYS" file. This one contains all the code to handle high speed SIO operations with the US Doubler 1050, Indus, Happy 1050, Speedy 1050, and XF551 drives. It also handles the standard speed SIO with all other drives and the standard parallel I/O (PIO) with devices such as the Multi I/O. 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 was 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 selected speed automatically turns over to be invalid anyway, you can still change it by hand using the SIOSET command (see chapter 4).

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

If you waive all that to go for the maximum free memory instead, you can load the SIO.SYS with the "/C" option (as "Classic"). This will load the old-fashioned SIO.SYS driver, as it was in SpartaDOS X 4.20. The only change to it is the capability to handle 512-byte sector devices.

If you do not need even the Classic SIO, you can use the SIO routines residing in the Atari ROM. To do that, pass the "/A" option (as "Atari") to the SIO.SYS driver. It will only occupy a minimum of memory, but the transmission parameters will depend, of course, 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.

CA2001.SYS Driver

Purpose
This is the fast speed SIO for California Access 2001 floppy disk drive.

Syntax
DEVICE CA2001 d:

Type
External - on device CAR:

Availability
As of SpartaDOS X 4.40.

Remarks
The driver does not allocate memory. It only tries to enable the fast serial protocol (38,400 bps) for the specified CA-2001 drive. The drive remains engaged as long as it is powered on.

INDUS.SYS Driver

Purpose
This is the high speed SIO installer for Indus drives. It is required for high speed operation with Indus drives and Happy drives.

Syntax
DEVICE INDUS

Type
External - on device CAR:

Remarks
This driver takes up no memory, it simply programs all Indus drives with the appropriate high speed code. (The Indus drives have high speed code already in them except that it is buggy code and does not work.) Once the drive is programmed, it will stay programmed until power is shut off on the drive. Thus, you do not necessarily need to program the drives every time you boot your computer. Also, you must install the "SIO.SYS" driver before the "INDUS.SYS" driver. This driver is required for Happy Drives.

RAMDISK.SYS Driver

Purpose
This is the SpartaDOS X ramdisk driver. You may select the drive and size of each ramdisk you install.

Syntax
DEVICE RAMDISK [drive][,nbanks]

Type
External - on device CAR:

Remarks
The default parameters for this driver are: drive 15 (O:) and all available windowed RAM less 4 banks for BASIC XE use (default is all banks on the Atari 800). You may install multiple drives by using different drive numbers and selecting the appropriate sizes. If you attempt to select more banks of RAM than there are left, it will use all available RAM. (To see the number of banks of RAM your system has, use the MEM command.) Note that each bank is 16 KB. Additionally, if the users 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 its drive number.

When the driver installs a ramdisk, it will automatically build the directory structure on the ramdisk. If you had previously installed the ramdisk with the same size and drive number before performing a COLD start without loosing power to the RAM, it will not be reformatted (the contents will be preserved). Therefore, performing the following commands will not destroy the contents of your *ramdisk.*

```
RAMDISK.SYS 8,4
COPY *.COM 8:
COLD (the system will reboot)
RAMDISK.SYS 8,4
DIR 8:
```
<all the COM files in drive 8 will be displayed>

A maximum of three ramdisks may be installed by RAMDISK.SYS. Any attempt to configure more will produce the error

```
RAMDISK not installed!
SIO device table full!
```

RAMDISK.SYS has no effect on Multi I/O ramdisks.

Some people think that the SpartaDOS X ramdisk is not as big as the 3.2 one or that SpartaDOS X does not recognize their whole RAM upgrade. This is not really the case. Since these thoughts are not uncommon, however, we will go into a little more detail here with system configuration.

At the beginning of chapter 8 the default system configuration is shown used if you do *not* have a CONFIG.SYS file on D1:. If you have 256K or more in an XL or XE computer, SpartaDOS X will automatically use one of the banks (USE BANKED) for DOS routines and drivers. This means that you have one less bank for your ramdisk, making it 16K smaller than it would be otherwise. You can write a custom CONFIG.SYS specifying USE OSRAM to allow you to use all available banks for your ramdisk. If you experience problems with your extended memory configuration to be used as a ramdisk, please e-mail the DLT team via the respective web page mentioned in the preface.

Also in that default configuration is "DEVICE RAMDISK". The default for RAMDISK.SYS is to use all available banks *beyond the four reserved for 130XE* programs and to assign the ramdisk to O: (see RAMDISK.SYS). You can change this, too, in a custom CONFIG.SYS file by specifying the drive number and number of banks. To override the reserving of the four banks, you *must* specify the number of banks in the "DEVICE RAMDISK" statement.
On a related topic, if you hold down the OPTION key when booting the computer, any CONFIG.SYS on disk 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.

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 is naturally not able to write data under itself. The PBI.SYS solves this (rare) problem. It also somehow speeds up the transfers on systems, where there is only one PBI device present.

CAUTION: MAP and SIOSET commands don't handle PBI bus devices with a driver installed.

TIMEKEEPING DRIVERS

ARCLOCK.SYS Driver

Purpose
Driver for the ARC clock.

Syntax
DEVICE ARCLOCK

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).

RTIME8.SYS Driver

Purpose
This is the R-Time 8 driver for SpartaDOS X. Without this driver (or JIFFY.SYS) installed, the TIME/DATE commands give the SpartaDOS X revision time and date (which does not advance).

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.

Z.SYS Driver

Purpose
Provide a SpartaDOS 3.2-compatible Z: device.

Syntax
DEVICE Z [/IS]

Type
External - on device CAR:

Availability
As of SpartaDOS X 4.40.

Remarks
A "Z:" device is created in the OS handler table, which offers a simple interface to SpartaDOS timekeeping functions, accessible e.g. from a BASIC program. The device is compatible with the driver existing for SpartaDOS 3.2 (ZHAND.COM there), so the old software using that should have no problems accessing the desired functions under SpartaDOS X anymore.

The 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

The correct procedure to read the time is as follows:

```
10 OPEN #1,4,0,"Z:":REM open for reads
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 writes
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$(13):REM reserve at least 13 bytes
15 OPEN#1,4,0,"Z:":REM open for reads
20 REM read formatted time
25 XIO 32,#1,4,0,"Z:"
30 INPUT #1;TIME$
35 PRINT TIME$
40 CLOSE #1
```

Z.SYS requires a hardware clock driver to be loaded first: RTIME8.SYS, ARCLOCK.SYS, JIFFY.SYS or any other compatible driver.

JIFFY.SYS Driver

Purpose
This is the jiffy clock driver for SpartaDOS X. Use this clock driver if you don't have an R-Time 8 or Atari Realtime Clock in your system. Without this driver (or RTIME8.SYS or ARCLOCK.SYS) installed, the TIME/DATE commands give the SpartaDOS X revision time and date.

Syntax
DEVICE JIFFY

Type
External - on device CAR:

Remarks
If there is already a clock handler installed (such as RTIME8.SYS) this handler will not be installed.

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 one of the joystick ports. The first parameter is the port number to be used (1 or 2, default is 2). It must have a monitor of its own. After installation, anything printed to the E: or CON: devices will be printed to the 80 column monitor through the XEP80. 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, which 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.

It should be noted 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, sending their output instead to the regular computer display. MENU.COM and FORMAT are good examples of this. Other programs use a combination of the two, such as the terminal programs 850 Express! 3.0 or BobTerm 1.2x. The actual online part of those will work properly on the 80 column screen, but the menus will show up on the 40 column screen. For these reasons it is a good idea to keep the 40 column monitor attached and running if space permits. A 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.

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 ON" at the DOS prompt, and then hitting the RETURN key, and later disable it similarly with "CON 40 ON".

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 8k of the main memory, having set 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, however, may quite happily be 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: while it still can consist maximum of three physical screen lines, a physical line, however, consists now of 64 characters instead of 40 – and that sums up to 192 characters per logical line.

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, however, can be less or more 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, and so 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) 1)GRAPHICS 8: this is the 320x192 bitmap mode with text window. In this mode, just as in the GRAPHICS 24, the BASIC's POSITION keyword is effective on the graphic cursor only. The text cursor position can be controlled through OS variables TXTCOL ($0291) and TXTROW ($0290), for the x and y coordinates respectively. In this mode you can safely both print text and draw graphics, because the screen driver maintains two separate sets of coordinates for the text and graphic cursors.

Note: The "text window" is not limited to the bottom three lines of the screen, but it 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.

CON80.SYS Driver

Purpose
This provides an 80 column display.

Syntax
DEVICE CON80

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 an 80x24 text console in software using the 320x192 graphic display. Its operation is very similar to the CON64.SYS driver described above, so this section will only discuss differences between them.

After installing, the system defaults to the standard 40x24 text mode. While in the Command Processor, you can enable the 80-column text mode by typing "CON 80 ON" at the DOS prompt, and then hitting the RETURN key, and later disable it similarly with "CON 40 ON".

Emulating 80-column text on a 320-pixel wide display requires much less calculations than for 64 columns. This is why the CON80.SYS is shorter than its 64-column brother, you can also find it being slightly faster.

The driver, however, may quite happily be used in BASIC and in your own programs. See "Using CON: drivers in own programs" (Chapter 6).

The CON80.SYS driver requires an XL/XE computer equipped with a 130XE-compatible memory extension.

KEYBOARD DRIVERS

CAD.SYS Driver

Purpose
This provides a "soft reset" procedure.

Syntax
DEVICE CAD keycode repeat ON|OFF

Type
External - on device CAR:

Availability
As of SpartaDOS X 4.40.

Remarks
Some programs, such as Disk Communicator 3, do not offer an option allowing the user to return to the DOS. The CAD.SYS solves this problem: when it is necessary to "kill" a running program, it is enough to press a defined key combination, and the system returns to the Command Processor, closing all the files and cleaning up everything it can.

The *keycode* parameter is a keyboard scan code of the key combination, which activates the "soft reset". The recommended values are:
$E7 (CTRL/SHIFT/INVERSE) or $CC (CTRL/SHIFT/RETURN).

The repeat parameter defines, how many times in a row the key combination should be pressed to activate the procedure. A zero means 256 times.

The last parameter decides whether the keyboard should generate upper (ON) or lower (OFF) case letters by default.

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 the following:

EXT,PROGRAM [,PARAMETERS]

where:

EXT – three-letter 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: 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.
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 for you automatically, you do not even have 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

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, and this can turn up 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.

Note: For best results, you should load the ENV.SYS before SPARTA.SYS, when the DOS is configured to USE BANKED, or after SPARTA.SYS otherwise.

The driver requires an Atari computer equipped with at least 128 KB RAM.

DOSKEY.SYS

Purpose
Command line extension.

Syntax
DEVICE DOSKEY [d:][path][fname.ext]

Type
External - on device CAR:

Availability
As of SpartaDOS X 4.42.

Remarks
DOSKEY.SYS, when loaded, 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 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.

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.

Additionally, the DOSKEY makes it possible to type more than one commands in one command line. You must use the '&'-character as the separator, for example:

```
CD DATA & DEL *.* & CD .. & RD DATA
```

This will execute these four command 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. You define them 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 command proper, which 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 EOL-terminated. 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.

DOSKEY.SYS requires an ATARI computer with at least 128 KB of RAM.

INIDOS.SYS

Purpose
Re-starting SpartaDOS X.

Syntax
none

Type
External - on device CAR:

Availability
As of SpartaDOS X 4.40.

Remarks
Executing the command COLD /N in the Command Processor, or causing a cold system restart while in BASIC or most application programs, deactivates the SpartaDOS X cartridge completely. The reactivation is usually not possible without switching the power off and back on – and this in turn, for example, invalidates ramdisks.

This problem can be solved using the INIDOS.SYS program. Copy it onto a disk, where the computer can be booted from, and then type in the command BOOT INIDOS.SYS. When the SpartaDOS X is about to be reactivated after COLD /N, you only need to insert this disk in the boot drive and then reboot the system (without switching the power off).

A DOS limitations

A.	Number of disks (partitions):	15
B.	Logical sector size:	512 bytes
C.	Number of sectors per disk:	65535
D.	Disk size (B*C):	33553920 bytes (~32 MB)
E.	Total capacity (A*D):	503308800 bytes (~480 MB)
F.	Directory size:	32768 bytes (32k)
G.	Number of directories:	unlimited
H.	Number of entries per directory:	1423
I.	Number of files per disk (G*H):	unlimited
J.	File size:	16777216 bytes (16 MB)
K.	Number of files open at a time:	16
L.	Path length:	64 characters

B 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 to get some of these error codes or messages under different circumstances but not likely.

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 which 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 can 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 when entered, so this error usually occurs within programs.

130 $82 (Nonexistent device)
The device specifier you used 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 not open for input)
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 not open for output)
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 a 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 disk 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 can occur under the following conditions: 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. 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 floppy 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.)

142 $8E (Serial bus overrun)
This error indicates that your *floppy* 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 floppy 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 disk, this error indicates that a sector is bad. If you are writing to a disk, you either have the disk write protected or the sector SpartaDOS is trying to write does not exist on the *floppy* (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.)

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 a filename already existing.

152 $98 "Not binary file"
You attempted to LOAD or run a file that is not a binary load file. There a 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 RTIME8.SYS or JIFFY.SYS drivers 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 on page 8-3. 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 filename you entered has a bad character in it. The two most common places you will get this error 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 the SpartaDOS Toolkit to remove this entry to allow the directory to be deleted. See more in Appendix C, "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 "Filesystem 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.

Error Message Summary

128	$80	User break abort
129	$81	File already open
130	$82	Nonexistent device
131	$83	File not open for input
132	$84	No device handler installed (Bad CIO command)
133	$85	File not open
134	$86	Bad file handle
135	$87	File not open for output
136	$88	End Of File
137	$89	Truncated record
138	$8A	Device does not respond
139	$8B	Device NAK
140	$8C	SIO framing error
142	$8E	Serial bus 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

C Command Summary - Alphabetical

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

APPEND *pathname*

Append the given path at the end of the $PATH variable.
Type - External - on device CAR:

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

Creates and maintains file archives.
Type - External - on device CAR:

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; Type - Internal

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

Enters the *internal* BASIC in a XL or XE computer (1200XL has no internal BASIC).
Type - External - uses CAR.COM on device CAR:

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

Loads the given file into the given memory area starting at the given address.
Type - External - uses CAR.COM on device CAR:

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

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

CAR [/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.
Type - External - on device CAR:

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;
Type – Internal

CHKDSK [d:] [/XV]
Shows volume, free/total disk space, and sector size of the selected drive (or diskette).
Type - Internal

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.
Type - External - on device CAR:

CHVOL [d:]volname
This command changes the volume name on the specified drive.
Type - External - on device CAR:

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

CLS
This command clears simply the screen. Especially for batch files it is very useful.
Type - Internal

COLD [/CN]
Reboots the system (by doing a jump through $E477).
Type - Internal

COMMAND (The Command Processor)
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:

COMP [d:][path]fname1.ext [d:][path]fname2.ext
Compare the given files
Type - External - on device CAR:

CON 40|64|80
Enables and disables the 64- and 80-column console mode.
Type - External - on device CAR:

COPY [/DIMNQRV] [+A|H|P|S] [-A|H|P|S] [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.
Type – Internal

DATE

Displays the current date and allows you to set the date.
Type – Internal

DELTREE [d:][path]dirname

Delete subdirectory trees recursively.
Type - External - on device CAR:

DF [/A]

Display summary information about free space on all disks.
Type - External - on device CAR:

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

Displays a long formatted directory including byte size, date, and time.
Type - Internal

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

Displays a short formatted directory (Atari DOS type).
Type - Internal

DPOKE [$]location [$]value

Puts a two-byte value (a word) into the given address.
Type - Internal

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

Displays a file in HEX and ATASCII form.
Type - External - on device CAR:

ECHO ON|OFF

Enable or disable the "echo" in the Command Processor.
Type - External - on device CAR: - executed by COMMAND.COM

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

Enable text editor.
Type - External - on device CAR:

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;
Type - Internal

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.
Type - External - on device CAR:

FORMAT

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.
Type - Internal

FSTRUCT [d:][path]filename.ext

Analyze a binary file.
Type - External - on device CAR:

KEY ON|OFF

Installs a 32 character keyboard buffer and links an "internal" KEY command into your system (for turning the buffer on/off).
Type - External - on device CAR:

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.
Type – Internal

MAN [command|/?]

Starts the documentation viewer.
Type - External - on device CAR:

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

SIO.SYS control.
Type - External - on device CAR:

MDUMP [$]address [$]len

Display memory in hex and ATASCII.
Type - External - on device CAR:

MEM [/X]

Displays the current low memory limits of your system and the number of available banks of windowed RAM.
Type - Internal

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.
Type - External - on device CAR:

MKDIR [d:]path

Creates a subdirectory. Alias - MD & CREDIR;
Type - Internal

MNT (syntax not available)

To address the KMK/JŻ IDE or IDEa devices if connected.
The description of this program belongs to the documentation of the respective hardware and thus will not be discussed here.

MORE <<fname.ext

Displays the contents of the given text file. Type - internal.

PATH [path_string]

Causes specified directories to be searched for commands before searching the current directory.
Type - Internal

PAUSE

Suspends system processing and displays the message "Press RETURN to continue".
Type - Internal

PEEK [$]address or PEEK symbol

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

POKE [$]location [$]value
Changes the contents of a memory location.
Type - Internal

PROMPT [prompt string]
Change the system prompt.
Type - Internal

RDDUMP d: [d:][path]fname[.ext]
Dumps the contents of the specified ramdisk to the specified file.
Type - External - on device CAR:

RDLOAD [d:][path]fname[.ext] [d:]
Restores the contents of the specified ramdisk from the specified file, that was created with RDDUMP.
Type - External - on device CAR:

RENAME [d:][path]fname[.ext] fname[.ext]
Changes the name of one or more files. Alias - REN;
Type - Internal

RENDIR [d:][path]dir_name_old dir_name_new
This command allows you to change the name of a directory.
Type - Internal

RMDIR [d:]path
Deletes an empty subdirectory from the specified drive.
Alias - RD & DELDIR; Type - Internal

RS232
Loads the RS232 handler from a P:R: Connection or the Atari 850 interface.
Type - External - on device CAR:

RUN [$]address
Run the code at the specified address.
Type - External - on device CAR:

S2I (syntax not available)
To address the SIO2IDE device if connected.
The description of this program belongs to the documentation of the respective hardware and thus will not be discussed here.

SAVE [d:][path]fname[.ext] [$]address [$]address
Saves binary data from memory to disk.
Type - Internal

SET [var[=env_string]]
To display the values of all environment variables and, optionally, sets an environment variable to a specified value.
Type - Internal

SIOSET [d: [type [usindex]]]
SIO.SYS serial speed control.
Type - External - on device CAR:

SL
This command generates a list of SpartaDOS X symbols.
Type - External - on device CAR:

SORTDIR [d:][path] [/NTSpartaDOS X]
To sort filenames in directories by name, type, date or size.
Type - External - on device CAR:

SWAP [d,d]
To swap (Re-map) your drive configuration or to display the current drive map list.
Type - Internal

TD ON|OFF
Turns on and off a time/date display line on top of your screen.
Type - External - on device CAR:

TIME
Displays the current time and allows you to set the time.
Type - Internal

TYPE [+A|H|P|S] [-A|H|P|S] [d:][path]fname[.ext] [/P]
Displays the contents of a specified file.
Type - Internal

UNERASE [d:][path]fname[.ext]
Restores files previously erased (if possible).
Type - External - on device CAR:

VERIFY ON|OFF
Turns write verify on or off.
Type - Internal

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.)
Type - External - on device CAR:

XFCONF [d:] [/12345]
To select density in floppy drives.
Type - External - on device CAR:

D Miscellaneous Notes

Using Turbo BASIC XL with SpartaDOS X

Turbo BASIC XL (TBS) has achieved great popularity in the world over the past 23 years. Since TBS uses RAM under the operating system ROM in XL and XE computers, it is not compatible with SpartaDOS 2.x and 3.x. However, with the proper combination of hardware and configuration TBS will work well with SpartaDOS X (SDX).

Hardware Configuration

It is necessary to have an XL or XE computer with more than 64 KB to use TBS with SDX. The Atari 800 has no RAM under the OS, making the running of TBS on this computer impossible with any DOS. Unmodified XEGS, 1200XL, and 800XL computers are also unsuitable, since SDX uses the RAM under the OS (USE OSRAM), conflicting with TBS, or a large chunk of low memory (USE NONE), raising MEMLO too high to be used with TBS.

System Configuration

It is necessary to use a bank of extended memory (USE BANKED) to run TBS. 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 on a SpartaDOS format disk in D1: to be able to use TBS.

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

```
USE BANKED
DEVICE SPARTA
DEVICE SIO
DEVICE INDUS
DEVICE CLOCK
DEVICE JIFFY
DEVICE RAMDISK
```

You can create the CONFIG.SYS file with the ED included on CAR:, a word processor or by typing

```
COPY CON: CONFIG.SYS
```

from the command line, typing in the above lines (making sure you press the RETURN key after each line), and then pressing the 3 key while holding down the CONTROL key. 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 the TBS main program, compiler, or runtime file. Use

```
X TURBO
X COMPILER
X RUNTIME
```

Using AUTORUN.SYS files

SpartaDOS X will not automatically load and run a file named AUTORUN.SYS when booted. With batch files and the relocatable nature of the SpartaDOS X command processor, however, the need for having an AUTORUN.SYS file is eliminated. There are three major types of AUTORUN.SYS you are likely to encounter. Following are descriptions of these 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 will take control of the computer when run. To use these, simply rename them to a relevant name and type that name from the command processor. 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 processor.

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, simply rename it something like MENU.COM and type

```
BASIC /N MENU
```

from the command line. Renaming the file is not required.

Using Batch Files

Any of these programs or 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 disk in D1:, the commands in the list will be automatically 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 or XE computer you must use a custom CONFIG.SYS file to use BASIC XE. To create one, follow the directions in the first full paragraph on page D-2, substituting "USE OSRAM"for "USE BANKED" in the first line of the configuration. This configuration is just an example. You may modify it as you see fit 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 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 with attempting to use BASIC XE again. The best thing to do is to do a cold start with the COLD command to clear out the extensions if you plan to run other programs.

Using MAC/65 and DDT with SpartaDOS X

When the computer is powered up, the MAC/65 cartridge initializes several page 4 memory locations and *never sets them again,* even when the cartridge is entered cold. Because of this, if you enter internal BASIC with the MAC/65 cartridge installed *before* entering the MAC/65 cartridge, those memory locations will be cleared. MAC/65 will then not work properly. To avoid this problem and still be able to use both MAC/65 and BASIC, use the CAR.SAV feature and enter the MAC/65 cartridge before entering internal BASIC. This will save those memory locations and restore them when you enter MAC/65. You will not be able to use the SpartaDOS X "LOAD" command to load files into memory and examine them from DDT, since entering the cartridge will restore the previous contents of that area. Use the MAC/65 "BLOAD" command instead.

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.

Using AtariWriter Plus with SpartaDOS X

If you have 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 128K of RAM in your computer, the procedure is a bit more complex. You will need to prepare a boot floppy for AtariWriter Plus. FORMAT a disk in SpartaDOS format and create a text file named CONFIG.SYS file on it. The 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 in D1:. Remove this diskette and insert the AtariWriter Plus diskette. 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 use this for temporary storage.

Using DiskRx with SpartaDOS X

The SpartaDOS sector editor "DiskRx" does not recognize the new DD 512 density, thus cannot be used to edit disks formatted in this way. Since the source code is not available, there was no other solution for that, than to write a suitable disk editor from scratch. The program has been written. It is called **Eddy**, and can be downloaded from:

http://drac030.krap.pl/en-sparta-pliki.php

Another problem with the "DiskRx" is that the program checks the file system type on the disk, where it is loaded from, and refuses to work, if it is not a SpartaDOS file system. The extended SpartaDOS FS built on a 512 bytes per sector disk is not recognized as correct by this procedure, so "DiskRx 1.9" cannot be loaded from such a disk. There is a patched version marked 1.9a, where this problem is fixed. It still, however, *cannot edit the DD 512 disks*.

Using CleanUp with SpartaDOS X

The "CleanUp" program from the SpartaDOS Toolkit does not have a clue about the newly introduced DD 512 density and thus, cannot be used to check and repair the file system built on such a disk.

When using 512 DD it is recommended to download "CleanUp X 1.1", the new file system consistency checker for SpartaDOS X from

http://drac030.krap.pl/en-sparta-pliki.php .

E Glossary

The following is a list of terms and their definitions as they apply throughout this manual and/or in many other computer publications.

ADDRESS — A location in memory. An address may refer to a RAM or ROM storage location, a hardware register for an external device or processor, or a combination of these through bank selecting. For the Atari 8-bit, addresses range from 0 to 65535 ($0000 to $FFFF hex)

APPEND — To add to. To append one file to another is to add the first onto the end of the second. This is often used when loading device handlers.

ASCII — The American Standard Code for Information Interchange. This code uses seven bit data words (0 to 127 decimal, $00 to $7F hex) to define a standard character and control set. For example, the character S is represented by the number 83 ($53 hex).

ASSEMBLY LANGUAGE — Assembly language is a readable representation of the machine language in which the CPU is programmed. 6502 assembly language uses three letter mnemonics and operands to represent the numbers of the instructions and arguments. This assembly language source code is then translated by an assembler to the numbers that the 6502 can understand. Advanced assemblers allow the source code to include labels, macros (collections of often used instructions), and more.

ATASCII — The Atari 8-bit version of ASCII. ATASCII uses eight bits, providing codes from 0 to 255 ($00 to $FF hex) and displayable characters for almost every code. There are several differences between ATASCII and ASCII, the most notable being that ATASCII uses 155 ($9B hex) as an EOL (end of line) marker, while ASCII uses a 13 ($0D) for CR (carriage return) and 10 ($0A) as an LF (line feed). Complete listings of ASCII and ATASCII values can be found in most programming reference books.

BANK — A block of memory of specified size occupying a specific range of addresses. *Bank selecting is* the use of hardware registers to cause different banks of RAM, ROM, or hardware registers to occupy the same address space.

This is necessary because the 6502, the CPU of the 8-bit, can only address 64 KB (65536 bytes) of memory. SpartaDOS X makes extensive use of bank selecting, allowing internal memory up to 1 megabyte (1024 KB or 1048576 bytes!) to be used effectively and easily.

BATCH — Batch files are text files containing a list of commands to be executed consecutively.

BAUD — A unit of measure of serial data transmission, named after the French inventor Jean-Maurice-Emile Baudot. This is the number of code elements per second. While this term is used interchangeably with bits per second (bps) in common usage, it is not actually the same.

BINARY — The base 2 numbering system. Each digit of a binary number can be only a 0 or a 1. This is the numbering system used by all digital computers, since each digit may be represented by either the presence or the absence of voltage. Since binary numbers such as 01010011 can be tedious to work with, numbers are usually represented in decimal (83) or hex ($53) format. A *binary file is* one that consists of numbers representing instructions and data directly readable by the computer. On the Atari, binary files also contain load addresses.

BIT — A single binary digit.

BOOT — The initialization of the computer, caused either by turning on the computer or by executing a cold start. The only difference in the two is that it is usually possible with SpartaDOS X to preserve ramdisk contents after a cold start but not after losing system power.

BUFFER — An area of memory used as a temporary storage area for data. Buffers are commonly used for I/O (input/output) involving the keyboard, screen, disk drives, etc.

BYTE — A binary number consisting of 8 bits. Since the 6502 processes 8 bits of data at a time, most data in the Atari 8-bit is represented as one or more bytes. Each address points to one byte. These values are most easily referenced as a two digit hex number.

CENTRONICS A standard parallel interface named for the company that first used it. Almost all parallel printers use "Centronics" type ports with a 'standard' 36 pin connector. The 8-bit Atari can be connected to a printer with a "Centronics" port only with an appropriate cable and an interface providing a parallel port, such as the Atari 850, the P: R: Connection, the Printer Connection, or the Multi I/O.

CIO Central Input/Output. All communication with the screen, keyboard, and all peripheral devices may be handled through this part of the Atari operating system. The CIO is one of the things that sets the Atari 8-bit above other 8-bit computers.

COLD START To cause the computer to initialize as if power were removed and reapplied without actually turning the computer off. This is faster than cycling power and will in most cases allow ramdisk contents to be preserved with SpartaDOS.

COMMAND An instruction given to the computer from the user.

CP Command processor. This portion of the DOS environment provides the interface between the user and the DOS. The CP prompts the user, interprets the commands given, and causes the specified operation to be executed.

CPU Central Processing Unit. This is the main part of the computer, the part that reads and executes instructions. All programs must be translated into commands and data that the CPU can understand. The CPU for the Atari 8-bit computers is the 6502.

CRC Cyclic Redundancy Check. This is a two byte number produced by performing a complex mathematical operation on a set of data. CRC is used in many applications, such as file transfer protocols and the ARC program.

CURRENT DIRECTORY The directory assumed if none is specified. The default is the main, or root directory. The current directory may be

changed to any subdirectory on a disk with the CHDIR command.

CURSOR — The mark on the screen that points to the place where the next action will take place.

CYLINDER — Used interchangeably with track, most often with hard drives, since these have multiple surfaces and heads.

DATA — Information used or processed by a program.

DEBUG — To isolate and correct errors in a program.

DECIMAL — The base 10 numbering system. This is the numbering system used by human beings everywhere, consisting of numbers made of digits ranging from 0 to 9. While easier to understand, decimal numbers can be awkward to use for computer purposes.

DEFAULT — The value or condition assumed if none is specified.

DENSITY — Generally, this is the number of bytes in each sector of a disk. Single Density refers to 128 bytes per sector, while Double Density refers to 256 bytes per sector.

DEVICE — An input and/or output interface to the computer, whether an actual physically external device, such as a printer, or a part of the computer simulating an external device, such as the screen editor. A program may access the D:, E:, S:, R:, P:, C:, K:, (the disk, screen editor, screen, serial port, printer, cassette drive, and keyboard, respectively) and other added devices through the CIO. SpartaDOS X, through the "kernel", uses the DSK:, CAR:, CON:, COM:, and PRN: devices. These "kernel" devices may be accessed through the D: device from the CIO or independently through the command processor.

DIRECTORY — The list of files and subdirectories disk or in a virtual disk (such as a ramdisk or the CAR: device). If there are subdirectories on the disk, then it is the list in a given directory.

DOS — Disk Operating System. The program which manages disk I/O to and from the computer. In practice, most DOS types also add functions and features to the operating system of the computer in areas not directly related to disk I/O.

DMA — Direct Memory Access. Any part of the computer system that can directly address system memory is considered to be a DMA device. The CPU is, of course, a DMA device. The only other DMA device on the Atari 8-bit system is ANTIC, the graphics coprocessor.

DRIVER — A program that usually remains in the computer and handles a specific device or operation. SpartaDOS X includes, for example, a driver to allow it to read from and write to Atari DOS 2 format disks.

FILE — A collection of information, usually stored as a named unit on a disk or virtual disk device.

FIRMWARE — Software that is permanently encoded into ROM, preventing it from being changed or erased. SpartaDOS X is firmware, as are the computer's OS, the US Doubler, and any cartridge-based program.

FORMAT — To initialize a disk so that it may be used to store and retrieve information. Physically the magnetic media on the disk is structured into tracks, which are divided into sectors. Directory information is usually written to a disk after it is physically initialized as part of the format operation. The SpartaDOS X command to format a disk is called FORMAT.

HANDLER — An often memory resident program that handles a device. This usually patches into the CIO handler table to allow applications programs to access the device as it would any other. For example, most DOS types add a D: device handler. An R: device handler must be added for most serial I/O applications.

HARD COPY — Information printed on paper.

HARD DISK A high capacity disk drive. hard disks are usually sealed units, storing anywhere from 5 to 150 megabytes of information. Storing data on and retrieving data from a hard drive is usually many times faster than performing the same functions on a floppy disk, especially if the hard disk is connected to the computer through the PBI. Hard disks require the use of a hard disk controller, although some have the controller built into the drive assembly: A SCSI or SASI interface, such as the ICD Multi I/O, is required to connect the hard disk controller to the computer.

HARDWARE The computer, peripherals, and all related circuitry. Generally, anything that you can touch can be considered to be hardware.

HEADER Data at the beginning of a file that provides information about the type of file, how it should be run, and where it should be loaded.

HEX Short for HEXADECIMAL, the base 16 numbering system. It consists of numbers made of digits ranging from 0-9 and from A-F (representing 10-15). This is the easiest and clearest way to represent the eight and 16 bit binary numbers used in the Atari computer. To differentiate them from decimal numbers, hex numbers are usually preceded by a '$' character.

I/O Input/Output. This refers to communication between the computer and the real world, including all devices and peripherals.

ICD The company that has written and designed power peripherals and software for the 8-bit Atari since 1984, including the US Doubler, P: R: Connection, Printer Connection, Multi I/O, R-Time 8, SpartaDOS Construction Set, SpartaDOS X, FAST Hard Drives, and more.

IOCB Input/Output Control Block. A sixteen byte block of memory used to pass parameters to and from the CIO for I/O functions. There are 8 IOCBs starting at $0340, numbered 0 through 7. IOCB 0 is normally used for the

screen editor. A different IOCB must be used for each open device or file.

K — Short for the metric prefix 'kilo'. A kilobyte is two to the tenth power bytes or 1024 bytes.

KERNEL — The center of SpartaDOS X, responsible for I/O functions. The kernel provides I/O independent of the CIO and provides many new devices, such as CAR:, CON:, PRN:, and DSK:.

KLUDGE — A program or part of a program or a hardware assembly that produces the desired results but does so in a complicated and/or unnecessary way.

LANGUAGE — A program or development system that provides an easier or faster way to write a program. A program written in a programming language is then either translated into machine code (compiled) or is used as a set of commands for another program (interpreted) which then performs the desired tasks. ACTION!, BASIC, Pascal, and C are examples of popular languages.

M — Short for the metric prefix 'mega'. A Megabyte is two to the twentieth power bytes or 1048576 bytes.

MACHINE CODE — The program that the CPU reads and understands. All programs written in other languages must be translated into 6502 machine code (often called machine language) before they can be run. While the terms assembly language and machine language are often used interchangeably, they are not the same.

MEMLO — The address of the start of usable memory above DOS, memory resident handlers and other programs. The number is stored at memory locations 743 and 744 ($2E7 and $2E8). When using BANKED memory, SpartaDOS X provides the lowest MEMLO available, meaning that you have more programming space available.

MEMORY RESIDENT — A program that remains in memory after being run and continues to perform its task or tasks when necessary. DOS is memory resident, as are added device handlers. Most memory resident programs relocate to low memory

and protect themselves from being overwritten by raising the low memory pointer.

MODEM — MOdulator/DEModulator. A peripheral device that translates serial data from a computer into sounds that may be transmitted over telephone lines, allowing communication with similarly equipped computer systems at a distant location. Some MODEMs, such as the Atari 1030, XM301, and SX212 may be connected directly to an Atari 8-bit computer, but most require a serial interface such as the Atari 850 or the ICD P:R: Connection. *Hayes compatible* MODEMS accept a standard set of simple commands to perform a variety tasks, such as dialing, answering the phone, hanging up, etc.

MULTI I/O — An interface made by ICD that connects to the PBI and provides up to 1 megabyte of RAM, an RS232 serial port, a parallel printer port, and a SCSI/SASI compatible hard disk port. The MIO allows reassigning the logical drive units of physical floppy drives, hard drives, and ramdisks, so that the computer can be turned off and booted from any of these.

NIBBLE — Four bits, or one half of a byte. A nibble can be represented by a single digit hexidecimal number.

PATH — A list of subdirectory names describing the course from either the root directory or current subdirectory to a specific subdirectory.

PARALLEL — The transfer, processing, or manipulation of all the bits in a byte simultaneously by using a separate line for each. This is usually faster than serial. Most printers are parallel devices.

PBI — Parallel Bus Interface, the large connector on the rear of the 800XL, allows the 8-bit Atari to communicate quickly with powerful external devices, such at the Multi I/O board. The PBI can be duplicated on the 130XE by a simple adapter connected to the cartridge and the ECI (Enhanced Cartridge Interface) ports.

PERIPHERAL DEVICE — Literally, a device on the periphery. In the computer world, a *peripheral device,* or just *peripheral, is* hardware added to the basic system configuration. Printers and MODEMs, for example, are peripheral devices.

PORT — A place of access to a system; e. g., the joystick port, the parallel bus interface port, the serial I/O port, printer port, cartridge port, etc.

PROGRAM — A set of instructions that cause a task to be performed by a computer. Programs must adhere to the order and conventions of the language in which they are written.

PROMPT — A signal to the user that some action may be required. The D1: or A: prompt with SpartaDOS X tells you that the computer is ready for input.

RAM — Random Access Memory. The storage area to which the computer may save and from which it may retrieve information. RAM will lose the stored information when power is removed.

RAMDISK — A specified area of RAM that simulates a disk drive. A handler or driver is used to make this memory appear to be a disk drive. Since they are actually RAM, ramdisks will lose their contents when power is removed. Except the new devices using SDRAM or Flash Memory of course.

REAL TIME — Relating to real world time. A real time clock uses the actual time. Real time can also refer to things that occur at the same time or at realistic speeds.

RELOCATABLE — A program that can be moved to different areas in memory and still operate properly. SpartaDOS X is relocatable. Most SpartaDOS drivers and handlers are self-relocating, usually meaning that they relocate themselves to low memory and move the low memory pointer MEMLO just above the code.

ROM — Read Only Memory. ROM is like RAM except that it can not be changed and will remain the same even after a loss of power. SpartaDOS X, all of the programs in CAR:, and the Atari OS are in ROM.

RS232 A standard serial communications interface documented by the Electronics Industries Association. Most MODEMS use an RS232 interface. The Atari 8-bit does not have an RS232 port, but one may be added with an Atari 850 interface or a P: R: Connection.

SCSI Small Computer System Interface. Many hard drives use a SCSI or SASI (Shugart Associates System Interface) bus. Up to eight SCSI devices may be connected to a SCSI port on a computer. The Multi I/O from ICD provides a SCSI port for 800XL and 130XE computers.

SECTOR The standard block of storage used on disks. Sectors on the 8-bit Atari may contain 128, 256 or 512 bytes.

SERIAL The transfer of information on one signal line, one bit at a time. Most MODEMs and all devices connected to the serial I/O port (SIO) on the Atari are serial devices.

SIO Serial Input/Output. All communication with devices connected to the serial bus of the Atari computer are handled through this routine. Devices on the parallel bus which simulate SIO devices (such as MIO ramdisks and hard drive partitions) are also accessed through the SIO. The Atari OS provides an SIO routine. SpartaDOS, however, uses its own SIO code.

SOFTWARE Programs, documentation, and data files that allow a computer to function.

SPARTA A powerful, disciplined, war-loving city-state of ancient Greece.

SUBDIRECTORY An additional directory on a disk allowing better organization through the grouping of associated files. Subdirectories are treated as entries in existing directories.

SYNTAX The order and wording of commands or statements.

TRACK A circular section of a disk surface. Each track is divided into sectors. A standard Atari 8-bit floppy disk is

formatted into 40 concentric tracks each containing 18 sectors.

TRUNCATED — Shortened. Usually used when the provided amount of data exceeds the expected amount and what can not be accepted is discarded.

VARIABLE — A symbol that represents a quantity that is changeable or has no fixed value.

VIRTUAL DISK — Something that appears to the system and user to be a disk drive but is not. Ramdisks, both internal and in the Multi I/O, and the CAR: device in SpartaDOS X are examples of virtual disks.

WARM START — A system reset that does not clear out all memory as a cold start does. Warm starts do reset several system pointers.

WILDCARDS — A symbol that is used as a substitute for one or more characters in a file or directory name to allow more than one file to be selected. '*' and '?' are the two valid wildcards with SpartaDOS and most others.

WORD — In the most common usage, a word is a sixteen bit or two byte number.

XIO — Extended Input/Output. A general I/O statement available in most Atari languages that allow CIO operations to be performed that are not supported by specific commands.

Index

SpartaDOS X

is the most advanced disk operating system ever designed for your ATARI 8-Bit computer. Its well known power has been brought to a new evolutionary level while still supporting the first generation machines ATARI 400 & 800 with at least 48 KB of memory.

This manual provides you with an in-depth look into the sophisticated world of the new SpartaDOS 4.42, 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, search paths, programming, and more. You may take full advantage of SpartaDOS X's outstanding power and features in little time and perform really complex tasks with ease and configure your system for optimum use.

Not only does SpartaDOS X cater to the entire 8-bit line, but also employs full use of extended memory up to 4 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, and a on-line help system, will support your usage a lot in getting more out of the system while using SpartaDOS X 4.42.

SDX
4.42

SpartaDOS X

User's Manual

December
2008