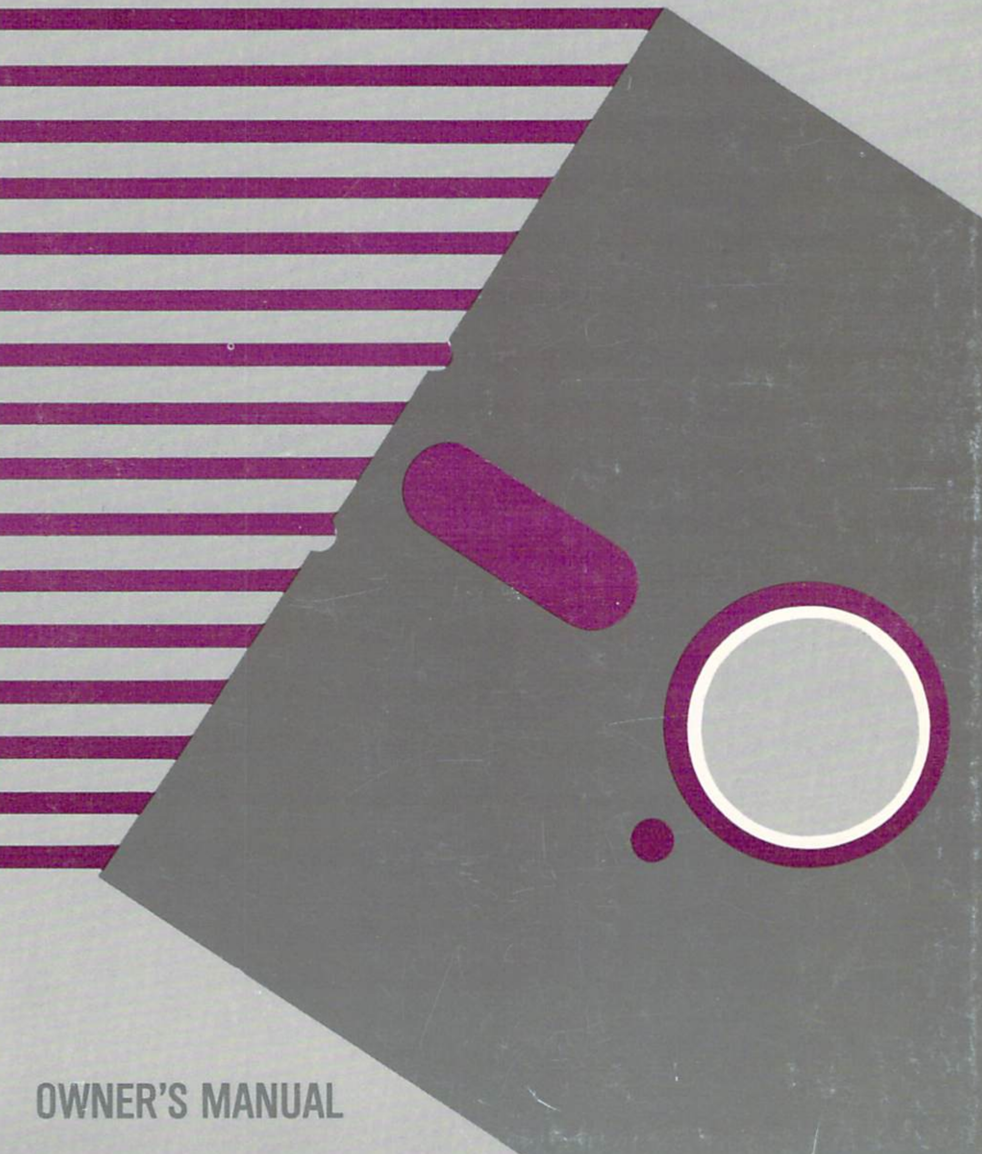
Sparta*DOS* Construction Set

Advanced Disk Operating Systems With Utilities for Atari 8-Bit Computers



OWNER'S MANUAL

The SpartaDOS Construction Set

Advanced
Disk Operating Systems
with Utilities
for Atari Computers

by ICD

Published by ICD, Inc.
1220 Rock Street, Suite 310
Rockford, IL 61101-1437
U.S.A.

PREFACE

The SpartaDOS Construction Set

What is a DOS? To some people a DOS is just for loading games. For others it is the framework for programming. Some even believe it is a silent manager that should never be seen. All of these are probably true. Different people want different things from a DOS just as they have different reasons for owning a computer. If you own an Atari 8 bit computer you are in luck! ICD has created the SpartaDOS Construction Set. This is one system, complete with useful utilities, choice of menu or command operation, even special memory efficient XL/XE versions with provisions for Ramdisk on the 130XE. SpartaDOS is the DOS for the future with support for any Atari compatible disk drive including future add on hard disks. It is the only DOS for 8 bit Atari computers that, as of this writing, supports single, dual **and** double density. SpartaDOS won't become obsolete just because a new drive comes out. Learn to use SpartaDOS **now** because it will last a long, long time.

What this Set will do for You

The SpartaDOS Construction Set is the cumulation of two major versions and several SpartaDOS types with many powerful utilities. This provides you, the computer user, with the building blocks for creating your own DOS diskettes. By working through this manual, you will learn the uses and requirements for: each DOS type, the commands, and the utility files. This should leave you with the fundamental knowledge needed to decide which DOS, if any, to use and which utilities are needed on which diskettes. After mastering the easy sections, you are invited to move on to the more technical chapters. There is enough meaty information in these sections to satisfy even the most voracious appetite. To the more experienced, we invite you to attempt writing some of your own SpartaDOS commands or utilities. This manual contains an abundance of new, useful information for everyone, from the beginner, to the most experienced programmer.

TABLE OF CONTENTS

CHAPTER 1 — INTRODUCTION

What is a DOS?

The Disk Operating System (DOS) is a special program which directs the internal operation of your Atari computer and disk drive. A DOS. . .

- manages the allocation and de-allocation of files
- provides a set of commands to interact with it
- provides a means of parameter passing to user programs
- provides a set of useful tools to aid in software development
- oversees the allocation of memory
- controls the flow of data in a system

Where is the DOS?

When your Atari computer is first turned on (booted), the computer's Operating System (OS) checks to see what devices are present. If a functioning Atari compatible disk drive is attached and set as 'D1:' (drive number 1), the computer will recognize the drive and try to read in a special program which should take control after it loads. This program is usually the DOS, and becomes a part of the computer's lower memory until the power is turned off. The DOS protects itself from being written over by other programs with a marker (MEMLO) which is placed just above its top of memory. Hopefully programs which then operate (run) with the DOS will obey this MEMLO marker and stay above it. So, where is the DOS? It was never in the disk drive. It is on a floppy disk and then read into the computer's memory. This is where a resident DOS remains, usually until the system is rebooted.

Power up Sequence and Why

It is important to power up your Atari computer system in the correct sequence or the disk drives will not be recognized by the system. Always turn drive 1 on before the computer, insert your DOS diskette into the drive and then power up the computer. The computer's operating system then recognizes the drive and starts loading the DOS. The other components in your system don't have any special requirements in the power up sequence, but generally the computer is powered up last. The power down sequence doesn't really matter as long as you take the diskettes out of the drives before turning their power off. Failure to due this may write bad information on the diskettes.

Different Uses of a DOS

Needs in a DOS vary from person to person. The following are some of the various uses for a DOS.

Storage

One common use for a DOS is to act as the storage device for another program. The AtariWriter and AtariArtist cartridges are good examples of this kind of DOS use. The system is booted up as usual but after the cartridge takes control, the DOS type commands are actually executed through the cartridge menu. The DOS is almost invisible to the user but still acts as the manager for disk storage.

File Management

File management becomes more important as system size increases. Things like subdirectories and time/date stamping have become invaluable in a well organized filing system. SpartaDOS is the only DOS that allows time/date file stamping on the 8 bit Atari computers. Subdirectories, like file folders, allow you to save different files under different catagories. Time/date stamping (when the file is created or rewritten) helps in maintaining constantly changing files and allows you to determine when it is time to discard others (expiration dates).

Binary File Loader

Binary files are machine language programs in file form with no protection built in. These normally can be executed (run) as command files under SpartaDOS, or they can be run under Atari DOS 2 with the 'L' menu command. LOGOMENU, our special menu program, makes binary file loading almost foolproof and provides a beautiful display (impress your friends) as well. This is a common use for a DOS and it is a good way to prevent the inexperienced user from damaging your valuable files by accidentally entering the wrong command.

Install Handlers

Handlers are special programs written to handle a device. An example of this would be a printer handler written for a specific printer, or a communications handler that provides a link to the communications line. The DOS is the most complex handler in the computer, but it will in turn, install other handlers as needed.

General Utilities

Utilities are included for housekeeping and programming functions. Commands like ERASE or RENAME will delete or rename a file. CHKDSK, RPM, and MEM are informational utilities which give important information about the condition of the system. MDUMP and OFF__LOAD are examples of informational utilities specifically for programmers. SpartaDOS was written in a way so that utilities can later be added without rewriting the DOS.

Miscellaneous Functions

SpartaDOS allows the rerouting of normal input and output of the system (called **redirection or diversion**). It also provides a standard for transferring information from one system to another.

What all this Means to You

We are providing all this information in the hope that some of you will read ahead to gain a better understanding of computer systems and someday, if not already, become the new computer literates.

CHAPTER 2 — AN OVERVIEW OF SPARTADOS

General Terms Used Throughout

The following is a list of standards used throughout the manual.

- The **ESC key exits most of the external commands** in SpartaDOS. Commands such as DUMP require that you use the BREAK key.

- A '<return>' means to press the RETURN key in our early examples. You may assume that a RETURN will terminate your input except in special cases (such as in INIT when single letter or number responses are required).

- The apostrophe or single quote mark is often shown at the beginning (') and end (') of a command or filename when written into general text. These are used as separators as in the example 'D3:INIT <return>'. The quote marks are not to be typed ; you would just enter D3:INIT and then press RETURN.

- Many commands require that you enter an address or an offset. It is safe to assume that **hexadecimal (HEX)** values should be entered. Hexadecimal is a base 16 numbering system which uses the digits 'A' through 'F' to represent decimal values of 10 through 15. If a number is preceded by a '$' in this manual, it is a HEX number. **Do not** type the '$' before a hexadecimal number with SpartaDOS commands.

- Many commands have restrictions as to what DOS and what format on diskette is involved. In the following examples 'n' refers to the major version number which will be 1 or 2. In these and all other descriptions, 'x' refers to the current revision level of that version. Here are some sample phrases and what they mean:

  CP version n.x
  The 'CP' stands for command processor. For internal commands, this indicates that the command processor understands the command. For external commands, this indicates that the command will interface to that version of SpartaDOS correctly. SpartaDOS version 1.x lacks many of the internal functions that version 2.x has. Thus if a command (such as MENU) uses a new internal function, it will not work with version 1.x.

Version n.x diskettes
Some commands under CP version 2.x (like LOCK and BOOT) will only operate on diskettes formatted by XINIT. The data table on sector 1 of SpartaDOS diskettes is slightly different between versions, thus the distinction is made (note that the major version is always in the table). **XINIT creates version 2.x diskettes, and INIT or FORMAT create version 1.x diskettes!**

Atari DOS 2 diskettes
This refers to any diskette formatted under Atari DOS 2 **or by the AINIT command.** Commands such as CWD, CREDIR, BOOT, etc. don't have meaning on this type of diskette since there are no subdirectories on these diskettes. Also note that SpartaDOS 1.x does not directly handle Atari DOS 2 at all, whereas SpartaDOS 2.x has an extended Atari DOS 2 handler built in. (SpartaDOS 2.x can read, write, and run Atari DOS 2 formatted diskettes in both single and double density).

- When the syntax of a command is given, several symbols are used to represent certain parts of the command. The following is a list of these symbols.

fname	This is the **filename** without an extension, thus it is from 1 to 8 characters in length.
.ext	This is the **filename extension**, thus it represents from 0 to 3 characters.
path	This is the complete directory path from the current directory to the desired directory. It **does not** include the filename.
[. . .]	This indicates that whatever is inside the brackets is optional. Do not type the brackets.

SpartaDOS Terms

The following are terms that are often used to describe SpartaDOS formatted diskettes.

Volume Names

When formatted, SpartaDOS diskettes are given a volume name. Each diskette should be given a unique volume name such as 'Games__1', 'Games__2', 'WP__1', '000243', etc. Volume names can be from 1 to 8 characters long and may include any of the 256 possible numbers, characters, or symbols, available on the Atari keyboard.

Version 1.x diskettes must have unique volume names, otherwise severe problems may occur!

SpartaDOS uses a sector buffering system quite different from Atari DOS 2. Whenever a sector is to be read, SpartaDOS first checks to see if it is in a buffer. When you change diskettes in a drive, there is no way for SpartaDOS to gain knowledge of this, other than to read a particular sector and compare a certain region to what it used to be. Thus, whenever a file is opened, the first sector is read and volume names are compared. If they are different, SpartaDOS will update its copy of the volume name and abort all sector buffers containing information about the previous diskette. If the old and the new diskettes have the same volume name, SpartaDOS will not know there is a new diskette in the drive and consequently the new diskette is in danger of being updated with bad information. Even though version 2.x diskettes have extra protection (random and sequence numbers), if they are used under version 1.x SpartaDOS, the extra protection is not used.

Directories

The diskette is broken up into directories, which may contain up to 128 files. The root (base) directory is named MAIN. Other directories (which are called subdirectories) can be created under MAIN. The same rules apply to both subdirectory names and to filenames except that the subdirectory names show up in the directory listing with '<DIR>' after the name and have special commands to create and delete them. Subdirectories may be nested under other directories with no limits restricting the total number of directories other than practicality and disk space. Paths are used to describe the connection from one directory to another.

The Current Directory
The current directory is the directory that you are presently in. If no path is given with a command or filename, then the current directory is used. The CWD (change working/current directory) command selects a new directory to be the current directory.

The MAIN Directory
The root directory (MAIN) is a special directory. Unlike subdirectories, it cannot be deleted. Whenever DOS is re-initialized (by RESET or by a new diskette being placed in the disk drive), SpartaDOS forces MAIN to be the current directory. It is good practice to keep any external command files in the MAIN directory. When an external command is used while you are in a subdirectory, SpartaDOS scans the subdirectory for the file. If it is not found there, SpartaDOS then checks the MAIN directory. Note that this is a process performed internally by SpartaDOS and is not a function of the command processor. The trigger for this action is simply the act of opening a file in read-only mode. Thus, this will work from BASIC, external commands, and any user application program.

Subdirectories
All directories other than MAIN can be thought of as subdirectories. SpartaDOS uses the tree directory structure, where the MAIN directory is the trunk and each subdirectory can be thought of as a branch (which in turn may have branches).

The path can be used in various DOS commands to specify which directories act as source and/or destination for the command. A '>' at the beginning of the path forces a start at the MAIN (root) directory. A '<' moves up the tree one directory (to the parent) and a directory name within the path selects a branch using '>' as a place holder between directory names.

Command Processor

Instead of a menu, commands are typed into a command processor much in the way commands are typed into BASIC. The extension of '.COM' is **reserved for external command files**. The general syntax of an external command is:

```
[Dn:][path>]fname [parameters] <return>
```

Note that the '[Dn:][path>]' should not be used when entering an internal command. Also internal commands should be in all upper case, whereas external commands may be in upper or lower case.

The extension of '.BAT' is **reserved for Batch files.** Batch files may be invoked by typing 'fname <return>'. Do **not** type the extension in either of these cases (although it is legal under CP version 2.x).

Menus

SpartaDOS provides two menu programs. One is a binary loader (discussed in Chapter 10) and the other is a general command menu (discussed in Chapter 11). The general menu is included for people who are more comfortable with menu operation. Note that the general menu (MENU.COM) is to be used under CP version 2.x only.

Handlers and Drivers

Many handlers or drivers are provided on your SpartaDOS diskettes. These load into memory and become resident once loaded (by linking into vectors and moving MEMLO up). Some examples of these are: RS232, AT_RS232, KEY, MENU, and RD130.

On Formatting Options

SpartaDOS allows many format options but your drive must have the specific hardware in order for the options to work, e.g. you cannot use double density format with the standard 810 disk drive or double sided format with any 810 or 1050 drive, etc. (These options may seem to write a format on an incompatible drive with no errors, but the format will not be fully functional). Once the format is written on a given diskette, the drive will automatically configure for that format type when trying to read or write to it. See Chapter 5 for more on formatting.

CHAPTER 3 — THE SYNTAX OF SPARTADOS

This chapter contains the details that Chapter 2 left out. These two chapters along with Appendix B are probably the most important in the manual. The rest falls into one of two catagories: 1) technical and programmer oriented information or, 2) detailed command descriptions. If you feel comfortable with SpartaDOS after reading this chapter, go ahead and try SpartaDOS; the best way to learn about a program is by experience.

Files

Unlike other Atari compatible disk operating systems, SpartaDOS supports subdirectories. This means that there are new rules for specifying which file you want to access. Files are specified by a *path* and a *filename* which taken together are considered a *pathname* and specify the location and name of a file. The definition of a pathname follows along with many examples to give you an idea of just how it works.

Filename Conventions

Filenames consist of a name and an optional extension separated by a period ('fname[.ext]'). Legal characters are as follows:

A..Z	(all letters of the alphabet)
0..9	(all numbers)
_	(underscore character)

The 'fname' part consists of from 1 to 8 characters and the '.ext' part consists of from 0 to 3 characters. Here are some examples of legal and illegal filenames, and if illegal, why.

TEST1.123	Legal name.
A FILE.LST	Illegal name. No spaces allowed.
ANOTHER.FIL	Legal name.
4TH.TRY	Legal name (numbers **not** restricted).
B_FILE.JN	Legal name.
PROG.BASIC	Legal name (the 'IC' will not be used).
DATA#	Illegal name. The '#' is not alphanumeric.
B	Legal name.

Actually, any filename is legal under SpartaDOS version 2.x, but once an offending character is encountered, no other characters are accepted. Thus, the second example would have the name 'A'. SpartaDOS version 1.x is much more fussy about filenames.

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 common extensions used along with the type of file it is used on.

.COM	Command file (a load and go file)
.BAS	BASIC SAVEd program
.TXT	An ASCII text file
.OBJ	An object code file
.SYS	A system file
.EXE	An executable file (like .COM)
.ASM	A machine language source file
.BIN	A binary data file
.DAT	A general data file
.PRN	A listing to be printed
.BAT	A batch file
.HEX	A hexadecimal coded file
.FNT	A font file (character set)
.LST	A LISTed BASIC program
.SRC	A general source file
.MUS	A music file
.DOS	A SpartaDOS module for the INIT/XINIT program

Wild Carding

Two wild card 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. This means that it will match any character in its position. The asterisk ('*') is like a "repeat until period" or a "repeat until end of filename" question mark. An asterisk can help in the speed of entering external commands. For example, 'OF*' can be specified instead of 'OFF_LOAD' as long as there are no other files that begin with 'OF'. The following examples illustrate the use of the wild card characters.

*.BAS	This represents the files in the directory that have an extension of 'BAS'.
.	This represents all files in the directory.

DATA??	This represents all files that begin with 'DATA' and have any combination of letters or numbers for the last two characters.
GR*.BAS	This represents all files that begin with 'GR' and have an extension of 'BAS'.
TEST.?B	This represents all files with the name 'TEST' and have any letter or number followed by a 'B' as the extension.

The internal commands DIR and COPY will supply a default filespec of '*.*' if none is specified. All internal and most external commands supply a device if none is entered. **Wild cards are legal in filenames if the file is to be read or matched, but are illegal if trying to save under the filename.** Examples of illegal usage when writing are:

```
SAVE FILE?.DAT 2000 3000      (or APPEND)
PRINT OUTPUT.*                (assumed D: device)
COPY E: TEXT*.FIL             (E: is not a directory device
                              thus TEXT????.FIL is the
                              name.)
```

Wild cards can be used with most commands that use a filename in the command line. One of the most common and time saving uses of wild cards is to execute external command files. Consider the following example:

```
DONKEY.COM
SPACE_IN.COM
GI_JOE.COM
MISSION.COM
FILE_MGR.COM
TELEPHON.COM
```

Any of the above files in a directory could be run by simply typing the first letter, and '* <return>'.

If 'DISMAL.COM' was included in the above example, then 'DI* <return>' would execute it; you should use 'DO* <return>' for 'DONKEY.COM'.

Caution: Wild cards are great time savers but can be very dangerous; read the warnings on using COPY and ERASE.

Directory Names

The same conventions apply to directory names that are used for filenames, however the extension (.ext) is **not** generally used on a directory name. In fact, most SpartaDOS utilities do not support extensions on directory names, but they will show up when doing a standard directory (SpartaDOS format). You may also use wild cards in directory names but the same restrictions still apply. That is, you may use wild cards when referring to a directory, but when creating (CREDIR) a directory, the name must be free of wild cards.

Path(s)

Since SpartaDOS can have more than one directory on each disk, it uses a path to describe the route from one directory to another. For our use, a path is the list of directories from the current directory to the destination directory. The '>' is a delimiter (place holder) between each directory name in the path. When you are not in the MAIN directory, you can also use one '<' for each directory to move backwards (to the parent directory). For ease of further explanation, directory names shall be referred to as '**dname**'. The '>', if used at the start of a path, moves the reference directly to the MAIN directory. The general syntax of a **path** is:

```
[>][dname>..dname]
```

where the optional starting '>' indicates to start at the MAIN directory and each 'dname' moves one directory along the path with '..' meaning "repeat until". An optional syntax of a path, which starts moving backwards (toward the parent directory), is:

```
<[<..<][dname>..dname]
```

where each '<' moves backwards one directory in the path. The rest of the syntax is the same as in the previous syntax. Suppose your diskette had the following directories:

(1)
```
Volume: TEST
Directory: MAIN

GAMES1              <DIR>   1-01-84  3:59p
TESTPROG  BAS       23717   4-05-85  2:45p
MODEM               <DIR>   3-09-85  1:18p
```

(2) Volume: TEST
Directory: GAMES1

```
ARCADE                  <DIR>   4-06-85  12:01p
BASIC                   <DIR>   4-06-85  12:04p
```

(3) Volume: TEST
Directory: ARCADE

```
MY__OWN    COM     12623    4-06-85  12:09p
FRIENDS    COM     8710     4-06-85  1:10p
LONER      DAT     3499     4-09-85  3:59p
```

(4) Volume: TEST
Directory: MODEM

```
XFER       BAS     23910    1-01-84  3:59p
RS232      COM     127      1-01-84  3:59p
```

(Note that the directory BASIC is not shown.)

For the following set of filenames, suppose that you are currently in the directory called GAMES1. The pathname given is how you would access that file. Note: If you were going to execute the command files, you would leave off the '.COM' extension.

```
RS232.COM       pn =   <MODEM>RS232.COM or
                       >MODEM>RS232.COM

FRIENDS.COM     pn =   ARCADE>FRIENDS.COM or
                       >GAMES1>ARCADE>FRIENDS.COM

TESTPROG.BAS    pn =   <TESTPROG.BAS or
                       >TESTPROG.BAS
```

For the next set of filenames, assume that you are currently in the directory called ARCADE. The pathname given is how you would access that file.

```
RS232.COM       pn =   <<MODEM>RS232.COM or
                       >MODEM>RS232.COM
```

FRIENDS.COM	pn =	FRIENDS.COM or >GAMES1>ARCADE>FRIENDS.COM
TESTPROG.BAS	pn =	<<TESTPROG.BAS or >TESTPROG.BAS

Note that the 'pn' in all the above examples is the full pathname. **For the rest of the manual, 'path' will refer to all but the filename and proceeding '>'** (if any) of the full pathname. Thus, the path refers to a specific directory, not the file in it.

The best way to become comfortable with pathnames is to experiment. Start creating subdirectories and keep trying new things until it becomes natural.

Command Types

The commands in SpartaDOS are of two types; **internal** or **external**.

Internal commands are directly understood by the command processor. They include commands such as DIR (directory), ERASE (delete file), etc. Most internal commands do **not** affect the program area (for BASIC, etc.). There are two exceptions: **COPY uses the program area as a buffer, and BUFS changes the low boundary of the program area. Both of these commands cause the cartridge to do a cold start (and thus wipe out any user program).**

External commands need to be loaded from disk each time they're used. They include commands such as TREE (list all directories), INIT (format a diskette), etc. **All external commands destroy the contents of the program area. These commands cause the cartridge to do a cold start (and thus wipe out any user program).** External commands interface to the command processor through a large data table. Therefore, they are able to accept command lines (filenames, numbers, and other parameters) for processing.

Default Drive

The default drive is the drive assumed when none is specified on a filename. **The default drive is only used when using the command processor!** To change the default drive, simply type the new device code (i.e. 'D2:') followed by a RETURN. The 'D' and the colon (':') **are required.** In the following example, the user input is in **bold:**

D1:**D3:** <return>

D3:**DIR** <return>

In the first line, the user changes the default drive to drive 3. On the next line, he/she does a directory of drive 3. Note that the normal syntax of the DIR command is: '**DIR [Dn:][fname[.ext]]**', but in the example the user did not type the '**Dn:**' (nor the 'fname.ext' — '*.*' was assumed).

Major SpartaDOS Version Differences

If you have booted SpartaDOS already, you have undoubtedly seen that SpartaDOS version 2.x is almost twice as big as version 1.x, so you ask why. Well, version 2.x contains everything in version 1.x plus the following:

- An enhanced Atari DOS 2 handler (lots of extras)
- Supports 8 disk drives (as opposed to 4)
- High Speed built in both 2.x versions
- 14 new internal command processor commands
- 8 new XIO functions
- Provides the user with an **extra 4K program area**
- Much better user error prevention
- 16 new external commands (over original 1.1 master)
- An Atari logo binary file loader
- A sophisticated DOS command MENU

There is one and only one catch, **you must use an XL or an XE Atari computer to run version 2.x!** (excluding the 600XL with only 16K memory and XL/XE computers with a modified OS chip installed.)

Directory Display Formats

The SpartaDOS file directory is revolutionary in the Atari world. SpartaDOS is the **only DOS that supports time/date stamping and gives file sizes in bytes (characters).** Only one other DOS for the Atari supports subdirectories (as of this writing), but none are as elegant or powerful as SpartaDOS.

The following is a typical directory listing:

```
Volume:    WRK__2.3B
Directory: MAIN

AT__RS232  COM  1863     2-12-85  6:38p
CHTD       COM  899      3-10-85  11:22a
UTIL            <DIR>    4-11-85  2:06p
CHVOL      COM  453      2-24-85  6:16p
DUMP       COM  1033     2-13-85  12:07p
RPM        COM  672      4-04-85  10:10p
XD23B      DOS  10729    4-06-85  1:17p
  577 FREE SECTORS
```

Notice that the volume and directory name are included in the directory header. This is an easy way to identify your diskettes. The time and date each file is created follows each file's (or directory's) name. The file sizes are expressed in bytes (rather than in sectors like Atari DOS 2). Subdirectories ('<DIR>') are easy to spot at a glance. **This type of listing is only given by the DIR command for SpartaDOS diskettes!** There is an alternate listing type that is as follows (for the same diskette):

```
* AT__RS232 COM 016
* CHTD      COM 009
* UTIL      DIR  002     (bold indicates inverse video)
  CHVOL     COM 005
* DUMP      COM 010
  RPM       COM 007
  XD23B     DOS  086
577 FREE SECTORS
```

Notice that this format has asterisks in front of some of the entries. An asterisk means that the entry is erase protected. That is, you can't erase or modify the file until it is unprotected (see the PROTECT and UNPROTECT commands). Also, the directory is indicated by an inverse 'DIR' as the file extension. The sector counts are derived from byte counts on SpartaDOS diskettes and are actual on Atari DOS 2 diskettes. **This type of listing is given if using the DIRS command or you are listing an Atari DOS 2 diskette directory!** Note that version 1.x does not have a DIRS command nor does it recognize the PROTECTed and UNPROTECTed status of files. Version 1.x does have a short form directory, but it is inaccessible through the command processor and the sector counts will show all zeros.

CHAPTER 4 — GETTING STARTED - STEP BY STEP PROCEDURE

This chapter is primarily an orientation to SpartaDOS. You will be taken step by step through formatting a diskette, displaying a directory, entering a small BASIC program, returning to DOS and doing a few file operations.

The Master Diskettes

The SpartaDOS Construction Set includes two 'MASTER' diskettes. Both are formatted in single density (90K). The disk with the **black label** has the version 2.x format along with the CP version 2.x DOS files, with commands and utilities that apply to SpartaDOS 2.x. Side A is the only side used on this diskette. The version 2.x DOS will only boot up on an XL/XE type computer. If you are using one of these machines for the following lessons, use the black labeled diskette. An error message will result if you try to boot this disk up in a non XL/XE computer.

The disk with the **grape label** was formatted on side A with SPEED.DOS, a CP version 1.x. This side also has the utility files which might be used with a version 1.x DOS. Side B is a demonstration of our binary file loader menu (LOGOMENU.SYS), running under NOCP.DOS, with several public domain games. Both sides of this diskette will boot up on any Atari 8 bit computer with at least 24K of RAM. If you don't have an XL/XE computer, use the grape labeled diskette for the following lessons.

Beginning Diskette Initialization and Duplication

You will be using BASIC, so if your computer is **not** an XL or an XE type computer, make sure you have a BASIC cartridge installed. XL/XE owners use the internal BASIC option. Next, boot the Master Diskette and wait for the 'D1:' prompt. Type 'XINIT <return>' if you are on an XL or XE computer, or 'INIT <return>' for all others. Now type 'N' for no DOS. You will be duplicating the Master Diskette, so all you are really doing is formatting a new diskette and giving it a volume name. **Now remove your Master Diskette**. Press a '1' as the drive to format, '1' for 40 tracks, '1' for single density, and type 'TEST <return>' for the volume name. If you have a US Doubler installed in drive 1, answer 'Y' in response to the next question, otherwise answer 'N'. Now insert a blank diskette into drive 1 and press RETURN. When the drive is done formatting the diskette, press the ESC key and the 'D1:' prompt should appear.

The next step is to duplicate the Master Diskette onto the newly formatted diskette. Re-insert the diskette you booted (the Master), and type 'DUPDSK <return>'. Answer '1' for the next two questions (source and destination drive are drive 1). Now press RETURN. When asked to insert the destination diskette, remove the Master Diskette and replace it with the one you just formatted. Now press RETURN again. If asked to insert the source diskette, replace the newly formatted diskette with the Master Diskette and repeat. When the copy is complete, press ESC to return to the command processor (a 'D1:' prompt should appear). You now have a backup copy of SpartaDOS. Now put the Master away so that it will not be damaged.

Brief Overview of a few Commonly Used Commands

Now that you have a backup, use it for the rest of this session. Type 'DIR <return>'. You should now see a file directory. Quite different from other Atari DOS's, isn't it. If you are using SpartaDOS 2.x, try the following: 'BASIC OFF <return>', 'CAR <return>'. Notice that it printed an error message. The command 'BASIC OFF' disables the BASIC cartridge and frees up that memory. Now type 'BASIC ON <return>' to re-install the BASIC cartridge. (Note that this only works with the internal XL/XE BASIC.)

Now type 'CAR <return>' (both versions) to enter the BASIC cartridge. You should now have a 'READY' prompt. Now type in the following BASIC program (end each line with a RETURN).

```
100 OPEN #1,8,0,"D:TEST.DAT"
110 FOR A = 1 TO 10
120 PRINT #1;RND(1)*100
130 NEXT A
140 CLOSE #1
```

Now type 'RUN <return>', you should notice that the drive starts spinning and the computer makes its usual beeping sound. When you get the 'READY' prompt, type 'SAVE "D:TEST.BAS" <return>'. This saves the program onto the diskette in the commonly used tokenized form. When you get the next 'READY' prompt, type 'LIST "D:TEST.LST" <return>' to save a text (ASCII) version of your program. Now type 'DOS <return>' to return to the command processor. Type 'DIR TEST.* <return>' to see the files you just created (notice the time and date — this is the default).

Now type 'TIME <return>' (this erases your memory version of the BASIC program). You will notice the top line has a time and date (which is rapidly ticking off the amount of time the computer has been on). When the clock has caught up, type 'SET <return>'. Now enter the current date and time as specified by the prompt, each followed by a RETURN.

Enter 'TYPE TEST.LST <return>'; you will now see a listing of the program you just typed in. Type 'TYPE TEST.DAT <return>'; this is the file your program just created. You can use the TYPE command to display these files because they are ASCII (text) files.

Now re-enter BASIC ('CAR <return>') and type 'LOAD "D:TEST.BAS" <return>'. List the program ('LIST <return>'), and resave it ('SAVE "D:TEST.BAS" <return>'). Exit BASIC again ('DOS <return>') and do another directory of the test files ('DIR TEST.* <return>'). Notice that 'TEST.BAS' has the current time and date on its entry.

Now type the following series of commands (end each line with a RETURN):

```
RENAME TEST.   * RANDOM.*
ERASE RANDOM.DAT
CAR            (you are now in BASIC)
LIST           (the program is still there)
RUN
DOS
DIR TEST.*
DIR RANDOM.*
TYPE TEST.DAT
```

Notice that there is only one 'TEST' file; this is the result of running the BASIC program. Also notice that it contains the current time. The BASIC program is saved under the name of 'RANDOM' in two forms. Also notice that the file 'RANDOM.DAT' does not exist (it was erased in the second line.)

Feel free to try more experiments at this point. The rest of the manual primarily describes the usage of the commands available in SpartaDOS. It is very important to just try new things to get a feel for the DOS.

Primary Commands

The remainder of this chapter contains the detailed descriptions of the DIR/DIRS, CAR, and BASIC commands. They are as follows:

DIR & DIRS Commands

Purpose

The DIR command displays the volume name and the specified directory name, lists files and subdirectories in the directory, the file sizes in bytes, the date and time the files were created, and the number of free sectors left on the diskette. The DIRS command lists the directory in a format similar to Atari DOS 2 (CP version 2.x only). The DIR and DIRS commands may be used to list all files matching a file spec pattern by using wild cards.

Syntax

DIR [Dn:][path>][fname[.ext]] or
DIRS [Dn:][path>][fname[.ext]]

Type and Restrictions

DIR is internal under CP versions 1.x and 2.x
DIRS is internal under CP version 2.x

Remarks

If no file spec is specified, all files will be listed (i.e. a default file spec of '*.*' is used). If no path is specified, the current directory is listed. Both DIR and DIRS work with SpartaDOS 2.x, while only DIR works with SpartaDOS 1.x. With version 2.x, DIRS displays a short form similar to Atari DOS 2 including the protected status (which DIR does not return). When reading an Atari DOS 2 directory with CP version 2.x, both DIR and DIRS give the same short directory result.

Example

```
    DIR
```

This command displays the entire current directory of the default drive.

```
    DIR D2:MODEM>XM*.*
```

This displays the directory range of XM??????.??? under subdirectory MODEM on drive 2.

CAR Command

Purpose
This command exits from DOS to a language cartridge.

Syntax
CAR

Type and Restrictions
Internal under CP versions 1.x and 2.x

Remarks
Since SpartaDOS is memory resident, any previously loaded cartridge program will remain intact when moving from DOS back to the cartridge, unless any external command or the COPY command was executed (or BUFS under CP version 1.x). If the latter is true, the program will be erased upon return to the cartridge.

Unlike other DOS's for the Atari, SpartaDOS gives immediate control to the DOS after power up. If you want the cartridge to come up automatically, create a STARTUP.BAT batch file on disk which contains the CAR command. (NOCP.DOS and XC23B.DOS give immediate control to the cartridge.)

SpartaDOS 2.x has built in error checking in the event no cartridge is present. With SpartaDOS 1.x, the CAR command will cause a system crash (lock up) if there is no cartridge present.

BASIC Command

Purpose
This command installs or removes the internal BASIC on the XL/XE computers.

Syntax
BASIC ON or
BASIC OFF

Type and Restrictions
Internal under CP version 2.x

Remarks
When the XL/XE computer is booted up normally with no cartridge plugged in, the internal BASIC is automatically installed taking up 8K of RAM. Holding down the OPTION key when booting will keep the internal BASIC disabled. The BASIC command will install or disable the built in BASIC and relocate the display memory as needed. This command can be included as the last command in a STARTUP.BAT batch file so you don't have to hold down the OPTION key. Note: the computer does a RESET (warm start) operation while executing this command. This causes the batch file to automatically close.

CHAPTER 5 — DISKETTE INITIALIZATION

The format disk command was very simple when the Atari 810 was the only disk drive available. The only choice was single density, single sided, and 40 tracks; one command was sufficient. As third party vendors developed more sophisticated drives, Atari owners suddenly had a choice. Percom, a leader in new products introduced double density drives, then double sided drives. SWP brought out the ATR8000 which could use almost any drive on the market. No longer was Atari DOS 2 sufficient for all these drives. Many Atari DOS 2 clones evolved to offer quick fixes but most just worked like Atari DOS 2 with a lot of patches. ICD has developed standard disk initialization commands which should eliminate disk format problems. These commands offer format menus which work with all available drives and can be upgraded easily for future drives without a major rewrite.

INIT & XINIT Commands

Purpose
These are the master formatting programs for SpartaDOS.

Syntax
INIT
XINIT

Type and Restrictions
XINIT & INIT are external under CP versions 1.x and 2.x
INIT will only create version 1.x diskettes
XINIT will only create version 2.x diskettes

Remarks
The INIT and XINIT programs are necessary since SpartaDOS can support many different drive configurations. These programs load SpartaDOS from '.DOS' modules which must also be on the diskette. The FORMAT command is a stripped down version of the INIT command which reads the DOS from an already existing version 1.x diskette with SpartaDOS on it.

Example

```
INIT or
XINIT
```

This program will display a menu of the possible SpartaDOS versions available on the disk along an 'N' option for no DOS; this is selected if you don't want SpartaDOS on the diskette but want it formatted. Assuming you selected a DOS, the correct module then loads into memory.

If using INIT, you are then asked if you want to 'modify default parameters'. You may select to write with verify, the default drive, and the number of buffers. These parameters are the defaults used when the new diskette is booted.
Next you will be asked which drive you want to format; valid selections are from 1 to 4 with INIT, and from 1 to 8 with XINIT.

INIT and XINIT then give a menu of tracks and sides. Normally you will use option '1' (40 tracks/SS) unless you are using an ATR8000 interface or Percom double sided drives.

The next choice, density, allows single density (128 byte sectors), double density (256 byte sectors), and 1050 double density (128 byte sectors).

'Volume name?' is the next question. You must enter a volume name (this should be unique to this particular diskette).

Next you will be asked if to use the UltraSpeed sector skew. Answer 'N' unless your drive is equipped with the US Doubler and you are using a high speed version of SpartaDOS. The US Doubler sector skew will be read slowly by a standard drive but 2-3 times faster in a US Doubler modified drive.

Now insert the diskette to be formatted and press RETURN. 'Diskette initialized. . .' will appear when finished. To format more diskettes, press RETURN; to leave this program, press the ESC key.

There are currently four versions of SpartaDOS 1.x and two versions of SpartaDOS 2.x. The uses for each version are as follows.

SpartaDOS 1.x Versions

There are presently two distinct SpartaDOS families. SpartaDOS 1.x was the first DOS family released by ICD. The 1.x versions work with all Atari 8 bit computers with at least 24K of RAM. Versions of SpartaDOS 1.x are generally limited to approximately 7K in size due to memory restrictions in the 400/800 computers. SpartaDOS 1.x versions are not Atari DOS 2 compatible, though files may be copied between DOS's using SPCOPY. All SpartaDOS versions now support UltraSpeed (high speed) I/O except for STANDARD.DOS (a 1.x version). The SpartaDOS 1.x versions are listed below.

NOCP.DOS

NOCP is a special high speed version of SpartaDOS to be used with AUTORUN.SYS programs like our LOGOMENU.SYS. This version functions much like Atari DOS 2; it has no command processor (NOCP means NO Command Processor). NOCP.DOS tries to load an AUTORUN.SYS file before it passes control onto the cartridge. Note that there is no equivalent of the DUP.SYS which Atari DOS 2 loads after the AUTORUN.SYS (if no cartridge was present). Some uses of this version are to load our LOGOMENU.SYS program (binary file loader), a printer handler for the AtariWriter cartridge, or the utilities disk with the Microsoft BASIC II cartridge. The I/O redirection (Batch files and PRINT command) is permanently disabled in NOCP.DOS.

NOWRITE.DOS

NOWRITE is a stripped down DOS with a very low MEMLO and short load time. Since it is a high speed version it will read in 2-3 times faster than a non-high speed version when used with UltraSpeed hardware such as the US Doubler. NOWRITE will run at normal speed when used with non-UltraSpeed drives. Anytime you attempt to write with a NOWRITE version you will get an error (usually 170). The main use of NOWRITE is for loading game files.

STANDARD.DOS

This is a SpartaDOS 1.x version without any UltraSpeed features. The MEMLO is about $300 bytes lower than the SPEED.DOS version. This is a RAM resident full powered DOS. Use this version for your regular DOS if you don't have a US Doubler modified 1050 disk drive and are not using an XL or XE computer.

SPEED.DOS

This is STANDARD 1.x version with the UltraSpeed code added. Use this version for your regular DOS if you have the US Doubler in a 1050 disk drive and don't have an XL or XE computer.

SpartaDOS 2.x Versions

These versions can be 11K or larger in size and use overlays beneath the OS ROMs in the XL/XE computers; therefor they will only work with the XL/XE computers. SpartaDOS 2.x versions are much more powerful than the 1.x versions; there are many more internal commands and they both support our UltraSpeed I/O. If you own an Atari XL/XE computer, it is advisable to use the 2.x versions for maximum benefit. They give you an extra 4K of usable free memory from BASIC as well as compatibility with most software. 2.x versions can also read, write and execute files directly from Atari DOS 2 type diskettes. Both SpartaDOS 2.x versions have 12 buffers built in so there is no need for the BUFS command. Both are CP versions; the only difference is whether priority is given to **D**OS or the **C**artridge upon boot. The SpartaDOS 2.x versions are listed below.

XD23B.DOS

This is the XD type SpartaDOS 2.x version and can only be used with XL or XE Atari computers. It is the most powerful DOS available for any 6502 based computer. This version has an extended command set, gives more free memory, and is more compatible than the SpartaDOS version 1.x. XD23B.DOS can read and write directly to and from other Atari compatible DOS's except Atari DOS 3 and OSS version 4. This version recognizes the STARTUP.BAT file when booted and priority is given to DOS (rather than the cartridge). For cartridge priority use the XC version below.

XC23B.DOS

This is the same as XD23B.DOS except AUTORUN.SYS is recognized when booted and control priority is given to the cartridge. All other features are the same as the XD version. The XC version will give a logon message before it starts loading the AUTORUN.SYS file. The BREAK key or SYSTEM RESET will abort the AUTORUN.SYS file if pressed just after the logon message is displayed. Control then goes to the cartridge (or DOS if you pressed OPTION and no other cartridge was installed).
This version can be used just like NOCP.DOS with programs such as the LOGOMENU program, AtariWriter, AtariArtist, etc. The XC version will only work with XL/XE computers and it **does not disable the I/O diversion** (Batch files and PRINT command). XC DOS does give you the complete command processor as in the XD version.

AINIT Command

Purpose
AINIT causes the drive to write an Atari DOS 2 style format. This command is mainly for compatibility with existing software, since SpartaDOS cannot be copied to and run under this format.

Syntax
AINIT [Dn:]

Type and Restrictions
Internal under CP version 2.x

Remarks
This will produce an Atari DOS 2 compatible format. The density is dependent upon the configuration of the drive as is normal with all Atari DOS 2 implementations. AINIT is the only internal format command and is supported with XIO 254. (See technical notes in manual for details.)

NOTE: This command will not produce a format with US sector skew which is needed for UltraSpeed I/O. SpartaDOS cannot boot from a diskette formatted in this way either. Use INIT for SpartaDOS 1.x or XINIT with SpartaDOS 2.x.

Example
```
    AINIT
```

The display will show:

```
FORMAT: Are you sure? Y/N
```

If you answer yes the drive will go ahead and format the disk with Atari DOS 2 type format.

FORMAT Command

Purpose
This command is used to format the diskette, create the directory structure, and optionally put DOS on the diskette.

Syntax
FORMAT

Type and Restrictions
External under CP versions 1.x and 2.x
FORMAT will only create version 1.x diskettes

Remarks
The FORMAT program allows many format densities, gives the user the option to put DOS on the diskette and to give the diskette a unique volume name. Once a diskette has been formatted, DOS cannot be put on the diskette without reformatting. The FORMAT program does not allow you to change boot defaults or choose many different SpartaDOS types; it reads the SpartaDOS with defaults from the diskette you use as the SOURCE.

Example
```
    FORMAT
```

The first question is whether to write DOS. If you answered 'Y', then you must insert a SpartaDOS source diskette of your choice into drive 1. After pressing RETURN, SpartaDOS is read into memory. The source can be any of the versions of SpartaDOS 1.x and the newly formatted diskette will retain the same defaults as the source.

The rest of the prompts are the same as the latter part of the INIT prompts.

Drive to format? (1-4 are valid)
Select number of tracks (usually #1)
Select Density?
Volume name? (1 to 8 characters)
UltraSpeed sector skew? (requires US hardware modification for high speed)

BOOT Command

Purpose
This command tells a SpartaDOS 2.x formatted disk to boot a particular program at startup.

Syntax
BOOT [Dn:][path>]fname[.ext]

Type and Restrictions
Internal under CP version 2.x

Remarks
The DOS loader on the first three sectors of each SpartaDOS 2.x formatted diskette, can load and run files in the same manner as a command file. Normally DOS is loaded, but actually anything could be loaded as long as it avoids the loader memory ($2E00-$3180). To change version 2.x DOS types on a diskette, first copy the new DOS file to the diskette, then use the BOOT command to force the new DOS to execute upon system boot. To create a binary boot diskette, use XINIT and select no DOS, COPY your binary boot file to the disk, then use the BOOT command which tells the loader the filename.

Example

```
    BOOT STAR.BIN
```

When this diskette is booted, it will immediately try to load and run the file STAR.BIN.

Ramdisk Commands

With the introduction of the 130 XE computer and the AXLON RAMPOWER 128 for the 800, users have discovered new applications for extra memory. One of the easiest to use is called a 'Ramdisk'. This is an electronic simulation of a disk drive using the extra memory as storage. The main advantages of this are great speed and two drive operations using only one physical disk drive. The main disadvantage is that the RAM memory is volatile which means that all memory is lost when the power goes down.

Purpose
These commands install a ramdisk device (electronic disk) in the place of a disk drive. Since these commands depend on specific hardware, the correct device must be present or an error will result. Note: CP version 1.x allows up to 4 drives and CP version 2.x allows up to 8 drives.

Syntax

RDBASIC Dn:	(XL/XE computer with internal BASIC on required)
RD130 Dn:	(Atari 130XE computer required)
RDAXLON Dn:	(Axlon RAMPOWER 128 in Atari 800 required)

Type and Restrictions
External under CP versions 1.x and 2.x
RDBASIC and RD130 **must be** installed under CP version 2.x only

Remarks
The Ramdisk is a simulation of a fast floppy disk. It is set up in 128 byte sectors and works with all the standard disk drive commands. The ramdisk handler is installed by entering the appropriate command along with the desired drive number. Any drive from 1 through 8 is valid with CP version 2.x ; drives 1 through 4 are the valid when using CP version 1.x. If there is already a drive in the selected location, the disk drive will be deselected (knocked out). Once installed, all standard drive commands will work with the ramdisk. If a ramdisk command is given and the required hardware is not in the system, an english error message will occur. Note: Do **not** install any one ramdisk to more than one drive number.

The Atari 130XE computer has 128K of RAM. The upper 64K can be accessed in 16K banks through an 'access window' between $4000 and $7FFF. The RD130 command allows easy access to this RAM and sets it up as a 64896 byte electronic disk (507 free sectors). This command works with CP version 2.x only.

The BASIC ramdisk works on all XL/XE computers and provides an extra 8K (7552 bytes) of RAM that was not used before (59 free sectors). This is only usable while the internal BASIC is installed. Holding down the OPTION key when booting or using the BASIC OFF command will destroy this ramdisk. The BASIC ramdisk area can be used as protected memory, scratch pad storage, and other uses which users will discover as SpartaDOS 2.x becomes familiar to the Atari community.

The AXLON RAMPOWER is a 128K board made for the Atari 800 which adds 8 - 16K banks of RAM to the system. These are seen in a window area from $4000 - $7FFF (similar to the 130XE but switched differently) and are switched through machine language handlers. RDAXLON sets the RAMPOWER board up as a 112K ramdisk. Since the RAMPOWER only works with the Atari 800 computer, RDAXLON will only work with CP version 1.x.

Examples

```
RD130 D5:
```

This installs the ramdisk as drive #5. If the computer is not a 130XE, an error message is generated.

```
RDBASIC D2:
```

The BASIC ramdisk is installed as drive #2.

```
RDAXLON D4:
```

You now have a 112K ramdisk as drive #4.

The RD130 Ramdisk may be setup as a utility disk under CP version 2.x with up to 64K of special utility files. These could be files like: MENU.COM, MENU.HLP, DUMP.COM, etc., which you may want to use but not keep on every disk. A STARTUP.BAT file could be created with the commands:

```
RD130 D2:
COPY *.COM D2:
```

When the computer boots this disk, the ramdisk is installed as drive #2 then all command files are copied to the drive #2 ramdisk where they will remain until powered down. All the commands can then be used on any SpartaDOS or Atari DOS 2.0 disk inserted into drive #1.

You may want to duplicate some files onto several disks. Copy the files to the ramdisk installed as 'D2:' then use:

```
COPY D2:*.* D1:
```

CHAPTER 6 — SUBDIRECTORIES

Subdirectories are an important feature of SpartaDOS, in fact, they were one of the major reasons for SpartaDOS in the first place. If you have never had subdirectories available to you before, you may be somewhat surprised at just how useful they are. This chapter describes that commands that directly manipulate or modify the SpartaDOS directory hierarchy.

?DIR Command

Purpose
To show the path to a specified directory. If no path is given as a parameter, the current directory path is displayed.

Syntax
?DIR [Dn:][path]

Type and Restrictions
Internal under CP version 2.x

Remarks
This command is normally used to show the current directory path. The path displayed is the path you would type after a CWD command to get from the MAIN directory into the directory you are currently in.

CREDIR Command

Purpose
This command creates a subdirectory under a specified drive and directory.

Syntax
CREDIR [Dn:]path

Type and Restrictions
Internal under CP versions 1.x and 2.x

Remarks
The directory to be created is the last directory in the path name. If no path is given, an error will occur. The path is in the format of 'NAME1>NAME2>NAME3' and indicates the route from the current directory to the directory to be created.

Example

```
    CREDIR D2:UTILITY
```

This command creates a subdirectory on drive 2 called "UTILITY".

```
    CREDIR GAMES>ARCADE
```

This command creates a subdirectory, ARCADE, on the default drive under the pre-existing subdirectory, GAMES.

DELDIR Command

Purpose
This command deletes an empty subdirectory from the specified drive.

Syntax
DELDIR [Dn:]path

Type and Restrictions
Internal under CP versions 1.x and 2.x

Remarks
The directory to be deleted must be totally empty before it can be deleted, and must be the last directory in the path name. Note that the MAIN (root) directory may not be deleted.

Example

```
    DELDIR GAMES>ARCADE
```

This command removes the subdirectory called ARCADE under directory GAMES only if it is empty, otherwise an error results.

CWD Command

Purpose
This command changes the current (working) directory on the specified disk.

Syntax
CWD [Dn:]path

Type and Restrictions
Internal under CP versions 1.x and 2.x

Remarks
The current directory is where DOS looks to find files whose names were entered without specifying which directory they were in. Also, the current directory is the base directory for relative pathnames.

Important: when a file is opened for read, the current directory is the first to be scanned for the file, but if it is not there, the main (root) directory is then scanned for the file. This is so that one may keep '.COM' files in the main directory and be able to access them from a subdirectory.

During DOS initialization, the current directory is reset to point to the main directory. Initialization occurs when the RESET key is pressed or when some application causes an initialization when it loads.

Remember that the current directory is displayed in the header of the expanded directory listing.

Note that the path can be substituted with '<' to move backwards in the path one directory (to the parent directory).

Examples
```
    CWD <
```

This command takes you backwards to the previous directory in the path.

```
    CWD D3:GAMES>ARCADE
```

This command takes you to the subdirectory called arcade on drive 3 under the subdirectory of GAMES.

TREE Command

Purpose
This command displays all the directory paths found on the diskette or under the specified directory, and optionally lists the files found in each directory in alphabetical order.

Syntax
TREE [Dn:][path] [/F]

Type and Restrictions
External under CP versions 1.x and 2.x

Remarks
The TREE command displays all path names found on the diskette when used from the main directory. If a path is specified, then all pathnames under that directory will be displayed. When used from a subdirectory, TREE will display all path names from that directory on. If the '/F' is specified, then all filenames in each directory will be displayed in alphabetical order after the directory path they're in.

Example

```
    TREE D1:MODEM /F
```

Subdirectory MODEM is displayed as the root directory and all filenames under that are displayed; then any subdirectories under MODEM are displayed along with the filenames under each of those. This continues until the last subdirectory and filenames are displayed.

CHAPTER 7 — DUPLICATION

This chapter describes most of the copy utilities that SpartaDOS provides. There are quite a few commands because of the different versions of DOS (XCOPY compared to SPCOPY) and drive configurations. (COPY is much more useful to those who have two drives or can use one of the Ramdisks provided.) Note that an XCOPY/SPCOPY type of copier is included in the MENU.COM program.

COPY Command

Purpose
COPY is an extremely powerful utility with many uses as follows. Copy one or more files from one device to another, and optionally give the new file a different name.

COPY can also copy files to the same diskette, however, the new file must have a different name or the destination directory must be different than the source. **Note that this command will not COPY a file between two diskettes using the same disk drive.** There is no provision to switch diskettes in the middle of the copy process. If a single drive copy is desired, use XCOPY, SPCOPY, DUPDSK, or the MENU program.

Under CP version 2.x, COPY with the '/A' option allows appending of two files (adding one file to the end of another).

You may also use COPY to transfer data between any of the other system devices, i.e. the Screen Editor, Printer, Keyboard, etc. (see the examples that follow).

Syntax
COPY d[n]:[path>][fname[.ext]] [dn:][path>][fname[.ext]][/A]

Type and Restrictions
Internal under CP versions 1.x and 2.x
Can be external in special cases under version 1.x
The '/A' option is only allowed under CP version 2.x

Remarks
The COPY command is the only command (aside from BUFS in CP version 1.x) that destroys memory. Thus, if you are writing a BASIC program, make sure that you save it before entering the command processor and using the COPY command. (The Ramdisks are useful for saving temporary information.)

The first file specified is the source file name. If none is given, a default filespec of '*.*' is assumed which will copy all files. The device for the source file must be given. The second file is the destination. If no filename is specified, a default filespec of '*.*' is assumed, which will copy without changing names.

You may use wild cards in both the source and destination filenames as well as in the extensions. If wild cards are used in the pathnames, the first directory match will be used. Multiple directories cannot be copied with one COPY command.

When using wild cards 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 filespec renames them by overriding characters in the source name when the destination name has characters other than '?' or '*' in it.

IMPORTANT: Only the device ID of 'D:' follows this convention since this is the only device that has directories. If a device other than 'D:' is used with the source filespec, then only one file is copied and the source filename is the source filespec, whereas if copying from the 'D:' device, the source filename is the filename from the directory that matches the source filespec.

Warnings for CP version 1.x only. In the example:

COPY E:*.* Dn:*.* **or** COPY E:

The destination filename is '????????.???' ('*.*' expanded out) since the Editor is **not** a directory carrying device, therefore, both the source and destination filespecs are the filenames. When saving a file named '????????.???', the first entry in the directory is matched **and no renaming process occurs on filenames written to the directory.** The end result is a file (called '????????.???') that is not erasable and one destroyed file (the first one).

In SpartaDOS 1.x, the internal COPY command resides in page 6 of memory. Occasionally another program might wipe this out and take page 6 for its own use. If this has happened, an error 170 will result when entering COPY. To continue use of the COPY command without page 6, an external file provision was built into SpartaDOS. Use the SAVE command to write the file COPY.COM onto the diskette with the offending programs. When the COPY command is called, a checksum is done to determine whether COPY is still intact. If not, the external file will replace it. The format to create this COPY.COM file is:

```
SAVE COPY.COM 600 6FF
```

CP version 2.x only.

When using CP version 2.x, the '/A' option allows the COPY command to append one file to another. The first file in the command line will be copied onto the end of the second file in the command line. (The file header from the second file will also be copied onto the end of the first file.)

The COPY command, aside from the obvious ability to copy and append disk files can also create batch files, print files on the printer, or allow typing directly to the printer.

Examples

```
    COPY D:*.PRN P:
```

This command copies all files from disk with an extension of .PRN to the printer.

```
    COPY E: D:INPUT.BAT
```

This command creates a batch file called INPUT. When this command is entered, the screen will clear and you may begin typing lines of text. When done, a CTRL 3 will signal the end of the file for the Editor and the data will be saved to the disk file.

```
    COPY E: P:
```

This example may be used for sending initialization sequences to the printer. The data you type will get printed on the printer.

```
COPY GAME2 GAME1/A
```

This example (only allowed under CP version 2.x) will append the file 'GAME2' onto the end of the file 'GAME1' on the default drive. If, before the command was executed, 'GAME1' was a 4000 byte file and 'GAME2' was a 2000 byte file, after execution 'GAME1' will be a 6000 byte file.

SPCOPY Command

Purpose
This command is used for single or dual drive file transfers between SpartaDOS and/or Atari DOS 2 compatible formats with few restrictions on density and number of tracks. This is the way to convert Atari DOS 2 files to SpartaDOS or the reverse of this (for CP version 1.x). Since translation is already built into CP version 2.x, use the smaller XCOPY with that version of SpartaDOS.

Syntax
SPCOPY

Type and Restrictions
External under CP versions 1.x and 2.x
XCOPY is suggested under CP version 2.x

Remarks
This utility program allows single or dual drive file transfers to or from SpartaDOS. SPCOPY is menu driven and the screen format is easy to follow. The screen is divided into four 'windows' as follows:

TOP	This window displays your path and filespec for your SOURCE filenames and the path of the destination directory. NOTE: no file renaming is performed so the '*.*' on the destination is unnecessary.
UPPER RT	This window displays the drive numbers selected for the source and destination diskettes.
LOWER RT	This window displays the command keys (and their function) along with the prompts used by this utility program.

LEFT	This window displays the selected directory from the diskette currently being read. You select files to copy by tagging them in this window.

Example
SPCOPY

The menu appears on the screen. The default setting is a single drive copy to and from the main directory. With the source disk in the drive, press **START to get the file list**. The directory is then displayed at the left with the arrow pointing to the current file. Press the **SPACE BAR to tag** the file or **SELECT to move on to the next.** Once all the desired files have been tagged, press **START to copy the files.** You will be prompted to swap disks as necessary.

SpartaDOS 1.x Restriction: Have different or unique volume names for each diskette since SPCOPY reads the volume name to determine if a different diskette is in the drive.

SYSTEM I/O ERROR: This is a general purpose error message given by SPCOPY when something goes wrong e.g. inserting the wrong disk when swapping source and destination, copying between two disks with the same volume name, etc.

XCOPY Command

Purpose
This command is used for single or dual drive file transfers between SpartaDOS and/or Atari DOS 2 compatible formats (with few restrictions on density and number of tracks). This is intended to be used with SpartaDOS 2.x since Atari DOS 2 format is recognized by the DOS (SPCOPY has Atari DOS 2 built in; XCOPY does not). If you are doing single or dual drive SpartaDOS to SpartaDOS copies, then this command is better even for version 1.x usage.

Syntax
XCOPY

Type and Restrictions
External under CP versions 1.x and 2.x
SPCOPY is suggested under CP version 1.x

Remarks
This command is identical to SPCOPY except for the following differences:

1. The Atari DOS 2 handlers are not built in. XCOPY assumes that the DOS can handle an Atari formatted diskette if necessary.

2. The file tagging has been improved so that you can see four files ahead of where you are currently tagging. (It scrolls the files before you reach the bottom.)

3. 100 files can be handled (SPCOPY can only hold 50 files).

4. SPCOPY re-initializes DOS at the beginning of each read and write pass; XCOPY never re-initializes DOS. This means that XCOPY is highly susceptible to volume names being the same (on version 1.x diskettes in particular). **PLEASE GIVE ALL YOUR DISKETTES DIFFERENT VOLUME NAMES!!!**

DUPDSK

Purpose
To duplicate an entire SpartaDOS diskette (except for volume name), using one or two disk drives. **Important: the number of tracks and the density on the source and destination diskettes must be the same or an error will result.**

Syntax
DUPDSK

Type and Restrictions
External under CP versions 1.x and 2.x

Remarks
DUPDSK is a disk copy program which will duplicate an entire SpartaDOS diskette including subdirectories while using one or two disk drives. This command **will not format or transfer the diskette volume name.** These must be created with a format program (INIT, XINIT, or FORMAT) since there are many possible variations in format. Also, the **destination format must be the same format type as the source format**, and the **destination diskette should not have any files on it as they will be overwritten!**

Example

```
DUPDSK
```

This command comes up with prompts for source and destination drives with 1 through 8 being valid drive numbers. You are then prompted to 'Insert Source diskette ?' if a single drive copy, or to 'Insert Source & Dest. Diskettes ?' if a two drive copy. Press any key (except ESC) to start the duplication and repeat as necessary if swapping diskettes in a single drive.

CHAPTER 8 — MAINTENANCE

This chapter contains the descriptions of commands used for erasing files, renaming files, and changing the volume name.

ERASE Command

Purpose
This command allows you to erase one or more files from a diskette and the specified disk directory. If no path is specified, then the file is deleted from the current directory. Wild cards can be used in the filespec.

Syntax
ERASE [Dn:][path>][fname[.ext]]

Type and Restrictions
Internal under CP versions 1.x and 2.x

Remarks
You may use wild cards in the file spec, however, use caution as one command can erase many files. If no filespec is given, an error will occur. Also, if a filespec of '*.*' is given, then all files will be erased and **no warning** will be given. Note that only files will be erased; any subdirectories will be left intact. To restore erased files, see the UNERASE command.

Note: Do not be alarmed if the free sector count seems off by one sector when copying, then erasing files. The directory and file maps are not assigned to specific sectors and will grow and shrink as necessary — but not always identically to a previous size or location.

UNERASE Command

Purpose
This command allows you to restore one or more files that were previously erased. If no path is specified, then UNERASE restores the files in the current directory. Wild cards can be used in the filespec. If a file can't be restored, UNERASE will indicate why.

Syntax
UNERASE [Dn:][path>][fname[.ext]]

Type and Restrictions
External under CP versions 1.x and 2.x
Caution: use a 1985 or newer UNERASE.COM version only!

Remarks
This command will restore files that have been accidentally erased, but only if they are still intact. If new files have been created since the desired file was last erased, then part of the erased file may have been overwritten and therefore lost fcrever!

Warning: UNERASE.COM files distributed before the release of SpartaDOS 2.x (dated in 1984), will totally destroy a CP version 2.x formatted diskette!

Example
```
    UNERASE *.*
```

UNERASE will map the current directory and then display the names of the files being restored as it encounters them.

NOTE: Occasionally the free sector count is decreased by one after an UNERASE. The reason for this is that the UNERASE command will increase the size of the directory file if the last sector is close to being full.

RENAME Command

Purpose
This command allows you to change the name of one or more files.

Syntax
RENAME [Dn:][path>]fname[.ext] fname[.ext]

Type and Restrictions
Internal under CP versions 1.x and 2.x

Remarks
Wild cards may be used in both filespecs. A device and path may only be specified on the first file name (the old name filespec). Filenames must be specified for both source and destination names, otherwise an error will occur. The rules for wild carding are described in detail in Chapter 4.

Example

```
RENAME FILEZ FILES
```

This command changes the name of the file on the default drive and default directory from FILEZ to FILES.

CHVOL Command

Purpose
This command is used to change the volume name on a diskette.

Syntax
CHVOL [Dn:]vname

Type and Restrictions
External under CP versions 1.x and 2.x

Remarks
CHVOL is used to change the volume name on a diskette. This can be useful if you change your mind on a volume label after it has been initialized. Note that you **may not** include spaces in the volume name, but any other character is legal. Only the first 8 characters will be used for the new volume name.

CHAPTER 9 — PROTECTION

Protection is an important feature in any DOS. It is quite easy to erase files using wild cards and not realize that a file you didn't want erased was lost. To solve this problem, a file protect status may be set on any file. If on, that file may not be erased until it is 'unprotected'. Also, a disk lock feature has been added which acts much like a write protect tab (or notch on 8 inch drives).

File Protection

File protection is accomplished by use of the PROTECT and UNPROTECT commands. These commands set or clear a bit in the file status. If set, that file may not be modified in any way. The following gives the command descriptions.

PROTECT Command

Purpose
This command protects (locks) files from accidental erasure.

Syntax
PROTECT [Dn:][path>]fname[.ext]

Type and Restrictions
Internal under CP version 2.x
SpartaDOS 1.x does **not** recognize the protect status

Remarks
PROTECT will help prevent accidental erasure of specified files. A write or erase attempt to a protected file will result in the message 'File protected' or error 164 from BASIC. Unlike Atari DOS 2, the RENAME function is allowed on protected files. Protected files will have an asterisk (*) before the file name when executing the DIRS command.

UNPROTECT Command

Purpose
This command unprotects files to allow you to erase of modify the files.

Syntax
UNPROTECT [Dn:][path>]fname[.ext]

Type and Restrictions
Internal under CP version 2.x
SpartaDOS 1.x does **not** recognize the protect status

Remarks
This is the reverse of the PROTECT command. Files must be UNPROTECTED in order to be modified or erased.

Disk Protection

SpartaDOS version 2.x has two commands that allow you to protect or unprotect on a whole diskette basis. This is similar to putting a write protect sticker on the diskette. However, there is one major difference; a write locked diskette may be written to by programs that do not use SpartaDOS file handling. The XINIT, INIT, and FORMAT programs are several examples.

LOCK Command

Purpose
This command locks the diskette to prevent accidental erasure. It is similar to the physical write protect tab which is put on the disk, but is strictly a software lock and only works when using SpartaDOS 2.x.

Syntax
LOCK [Dn:]

Type and Restrictions
Internal under CP version 2.x
SpartaDOS 1.x does **not** recognize the protect status

Remarks
The LOCK command has been added to SpartaDOS to allow write protection of the SpartaDOS 2.x diskette. The lock byte is physically written to the diskette where it will remain until the UNLOCK command is given. The status of LOCK ON or OFF can be checked with the CHKDSK command. When trying to write to a locked diskette while in SpartaDOS 2.x the message 'Disk write locked' will be displayed. Under BASIC it will be an error 169 ($A9).

UNLOCK Command

Purpose
This command unlocks a SpartaDOS 2.x formatted disk (See LOCK).

Syntax
UNLOCK [Dn:]

Type and Restrictions
Internal under SpartaDOS 2.x
SpartaDOS 1.x does **not** recognize the protect status

Remarks
This command updates tne SpartaDOS 2.x diskette to allow writing to it. UNLOCK is the reverse of the LOCK command. After executing the UNLOCK command, CHKDSK will show 'Write lock: OFF'.

CHAPTER 10 — LOGOMENU - STEP BY STEP

How to Create a Special Binary File Loader

Many Atari users have a collection of binary files, many of which are games. For this, special versions of DOS (NOCP and XC23B) and a menu program (LOGOMENU.SYS) have been created to load and run these files. Here are some of the advantages of using the NOCP/XC23B DOS in conjunction with LOGOMENU.SYS:

a) File protection - since there is no command processor in NOCP (similar to using DOS.SYS with no DUP.SYS in ATARI DOS 2) it becomes difficult to accidentally erase or write over a file without first booting up another version of SpartaDOS. Also, since LOGOMENU.SYS is only a load/run type of menu that you can't exit aside from rebooting; only reading from the diskette is performed.

b) UltraSpeed - both versions of DOS run in UltraSpeed mode as long as your drive hardware supports it and you select US sector skew when formatting. Otherwise, it will run at standard speed.

c) Size and Organization - LOGOMENU.SYS can handle up to 16 directories each with up to 64 files. It is arranged so that the SELECT key toggles the page of files within a subdirectory to be displayed, and the OPTION key toggles the subdirectory currently being displayed. When you choose a file to load and run, simply press the letter of the file.

d) Simple Operation - a double wide menu will display during operation. You press the letter or letters that correspond to the file and it loads and runs. To see more pages of files, use the SELECT/OPTION keys as described in (c). Once a diskette of files has been created (as to be described), no other steps need be taken other than turning the system on and selecting the file to run.

e) Compatibility - NOCP has a MEMLO about 1/4K above that of Atari DOS 2 and XC23B has a MEMLO about 4K below that of Atari DOS 2. All games that run under an Atari DOS 2 type menu should run under this menu.

f) Hidden Files - if you 'protect' a file (by the PROTECT command), that file will **not** show up in the menu. If you 'protect' a subdirectory, that entire directory will not be included in the menu. This way you may put several catagories (subdirectories) on one diskette, but only allow certain catagories to be displayed when used.

h) Deselects BASIC - LOGOMENU automatically deselects the internal BASIC on XL/XE computers, so there is no need to hold down the OPTION key when booting your LOGOMENU diskette.

Construction (for non-XL/XE Computers):

Initialize a disk with NOCP.DOS using INIT.

This is a version 1.x type SpartaDOS. Insert a diskette with the files INIT.COM and NOCP.DOS into your drive and boot up. If a batch file runs and pauses, press RESET to get the 'D1:' prompt. Type INIT and then press RETURN. The DOS menu will come up. Press the corresponding number for NOCP. After it reads the NOCP file it asks to 'Modify defaults?', press 'N' for no.

Then, drive to format is usually drive 1, tracks will be 1 for Atari drives; the other selections are for the ATR8000 and similar peripherals. The density menu gives a choice of 1) single (90K), 2) double (180K), and 3) 1050 double (130K). 810s only get number 1, 1050s can choose 1 or 3, and 1050s with the US Doubler can choose any of the densities.

The volume name should be unique to each diskette. You might want to use numbers or letters or both, as it is intended to help you keep track of your diskettes. US Doubler owners will type 'Y' (yes) for UltraSpeed sector skew, everybody else will type 'N' (no).

Now insert the diskette to be formatted into the drive specified and press any key to begin. After the diskette is formatted, it is a good time to repeat the procedure on any other diskettes you might want to initialize as games diskettes.

Copy LOGOMENU.SYS to the MAIN directory on each of your newly initialized diskettes and RENAME it to AUTORUN.SYS.

Find a diskette with the file LOGOMENU.SYS on it. You can use SPCOPY, XCOPY, or COPY if you have 2 drives. After the file has been copied to the destination diskette, RENAME it to AUTORUN.SYS. Example: 'RENAME LO*.* AUTORUN.*'. Note: This will be the only file (other than subdirectory names '<DIR>') stored in the MAIN directory on your games diskettes.

Create subdirectories for file storage on each of the destination diskettes. All files must be stored in subdirectories; not the MAIN directory.

Use the CREDIR command. You may only want to use one subdirectory on the disk if there will only be a few files on it. Name subdirectories to help organization (e.g. SPACE for space games, PACMAN for pacman type maze games etc.). Example: 'CREDIR PACMAN'. This writes a subdirectory called 'PACMAN' on the diskette under the MAIN directory.

Copy the files to your new diskettes under the desired subdirectories.

Use SPCOPY or XCOPY to copy the files. Under the destination file name you must put the subdirectory path in the proper format. Example: 'PACMAN>*.*'. This will only work if the subdirectory named 'PACMAN' is on the destination disk. Note: Do **not** copy any of your game files to the MAIN directory. 'AUTORUN.SYS' is the only file allowed under 'MAIN'.

Boot the new games disk and try it out!

SELECT scrolls the directory display up to show additional filenames if any. OPTION changes subdirectories if more than one is on the diskette. SYSTEM RESET reloads the directory. This is helpful in the event that you change diskettes in the drive. There should not be any cartridges installed although it is OK to leave the R-Time 8 Cartridge installed. Internal BASIC will be automatically deselected in the XL/XE computers.

Notice: The multicolor symbol displayed is the logo, trademark, and property of the Atari Corporation.

Trouble shooting:
The display comes up with READY or MEMO PAD/DIAGNOSTICS - 'NOCP.DOS' needs a file called 'AUTORUN.SYS' in order to initialize properly. Check the MAIN directory for the file 'AUTORUN.SYS'. Also, the BASIC cartridge must not be installed.

The display comes up with ERROR: NO SUBDIRECTORIES FOUND - You must have at least one subdirectory on the diskette.

The display comes up but doesn't show any filenames - There are no files stored under the subdirectory. Boot up a STANDARD or SPEED version of DOS, then put the games disk in and check the directories. 'DIR' will show the MAIN directory and 'DIR PACMAN > ' will show a subdirectory called 'PACMAN'.

Construction (for XL/XE Computers Only):

Initialize a disk with XC23B.DOS using XINIT.

Insert a disk with XINIT and the 2.x versions of DOS into drive 1 and boot up the system. Then follow the instructions for construction with 'NOCP' except you must substitute select 'XC23B' instead of NOCP. Also the 2.x versions show up as filenames under the MAIN directory; the 1.x versions remain hidden from view. The rest of the instructions are the same.

When trouble shooting 2.x version, you may also get the D1: prompt when either SYSTEM RESET or BREAK is pressed; if there is no 'AUTORUN.SYS' file in the MAIN directory, then either 'READY' or the 'D1:' prompt will appear.

CHAPTER 11 — MENU OPERATION

Do you still prefer the Atari DOS 2 menu over a command processor driven DOS? Well, SpartaDOS has a menu program too. But be warned, don't expect it to even resemble that of Atari DOS 2's menu. I suggest that you run MENU now and see what it looks like **before** you continue reading the command description. The description should make more sense once you know what the display looks like.

MENU Command

Purpose
This command gives you most of the features of the command processor but in a menu form. It is capable of single and multiple file functions.

Syntax
MENU [R][n]

Type and Restrictions
External on CP version 2.x

Remarks
If you type 'R' on the command line, the MENU program will remain resident. This means that you may go to BASIC and then type 'DOS' to re-enter the menu program. If 'R' is not specified, MENU will not be able to be re-entered once you exit it.

There is a help file ('MENU.HLP') the MENU uses when you ask for online help. The 'n' parameter sets the drive that this help file will be on. This gives you the ability to load the file into a Ramdisk and use it from the menu program. If 'n' is not specified, drive 1 is assumed. Note: if you specify both 'R' and an 'n', do not put a space between them (i.e. 'R5'). The operation of MENU follows in the next section.

The MENU Operation
Many of the MENU functions (e.g. copy, erase, protect, etc.) can do the operation on many files at once. This is done by tagging the files you want the operation to be performed on. The following key strokes are used:

up arrow	- This moves the 'select' cursor to the file above the current. If at the top, the cursor will move to the last file in the list.
down arrow	- This moves the 'select' cursor to the next file in the list. If at the bottom, the cursor will move to the first file.
space	- This toggles the tagged status of the file. The file is in inverse video if it is currently tagged.

The next step is to select the command or function you wish to perform. For this you must get the command cursor (in the bottom boxes) on the correct command. The commands are arranged in 5 banks of 5 commands. The following key strokes select the command:

OPTION	- This selects the next bank of commands. The cursor remains in the same position. There are 5 banks in all.
SELECT	- This moves the cursor to the next command (to the right). If at the end, it moves to the first command in that bank of commands.
right arrow	- This is identical to the SELECT key.
left arrow	- This moves the cursor to the last command (to the left). If at the beginning, it moves to the last command in that bank of commands.
1 .. 5	- The number keys 1 through 5 select a bank of commands. This gives an alternative to the OPTION key and allows you faster access to the row you want.
A .. Z	- The letter keys move the cursor to a particular command. This allows you to memorize letters for the most used functions for faster access. An attempt has been made to make the letters correspond to the function.
HELP key	- This gives you a small description of the command the cursor is on. To restore the screen after HELP, press the RETURN key (actually any key will work).

Once you have selected a function, you may perform it by pressing the **RETURN** or **START** keys (they are functionally identical). A description of each command follows. The corresponding letter command is given in parenthesis following the command name.

?Files (F)	- This command does a directory of a drive that you specify. This then becomes the source drive for copy and all other functions (that pertain) operate on the selected drive. When asked 'Which Drive?' answer with a drive number or RETURN for Drive 1.
Copy (C)	- This command will copy all tagged files, or the file the cursor is on if no files are tagged. When asked 'Dest Drive?', enter the destination drive number of the copy or RETURN for Drive 1. Next enter the path name of the destination directory or RETURN for the MAIN directory. When prompted to insert diskettes, press RETURN for the copy to continue.
Erase (E)	- This command will erase all tagged files, or the file the cursor is on if no files are tagged. No prompts are given so be careful.
Rename (R)	- This command renames the file the cursor is on to a name you specify. When asked 'Rename to?', enter the new name and RETURN.
Exit (Q)	- This command exits the menu program and enters the command processor. Caution must be taken when a cartridge is also enabled. Read the 'Other Notes' section at the end of this chapter **carefully**.
RunCar (B)	- This command exits the menu program and enters a cartridge if it is enabled. Caution must be taken when a cartridge enabled. Read the 'Other Notes' section at the end of this chapter **carefully**.
Load (L)	- This command loads the file the cursor is currently on. The file must be a binary file. The screen will be cleared before the file is loaded. The standard Atari DOS 2 INIT and RUN vectors are used. Once the file has run, press RETURN to re-enter the menu program (if the file doesn't take over).

Save (S)	- This command saves a binary file. You must enter the filename (and path), the start address, and the end address as requested.
Run (J)	- This command jumps to a machine language program. You may either specify an address, or press RETURN. In the latter case, the beginning address of the last file 'LOADed' will be used. The screen is cleared before the machine language program is entered. If that program allows, you may press RETURN to re-enter the menu program.
Exec/P (G)	- This command loads the file the cursor is currently on. But before it loads, you can give a command line to that file. This is how to use external commands under the MENU program. The screen is cleared before the file is run, and when done, you may press RETURN to re-enter the menu program.
XInit (I)	- This command loads the XINIT external command and runs it. This is how to format SpartaDOS version 2.x diskettes. To exit the XINIT program and re-enter MENU, press the ESCape key.
AInit (A)	- This command formats a diskette in Atari DOS 2 format. Enter the drive number when asked (RETURN for drive 1) and then any key when ready to format.
?Mem (M)	- This command displays the contents of MEMLO and MEMHI. Press RETURN to restore the display.
ChkDsk (Z)	- This command performs the CHKDSK command. Press RETURN to restore the display.
Help (H)	- This command provides help on the keys used to move the cursor and select a command. Press RETURN to restore the display.
Prot (P)	- This command will protect all tagged files, or the file the cursor is on if no files are tagged.
UnProt (U)	- This command will unprotect all tagged files, or the file the cursor is on if no files are tagged.

Lock (K)	- This command write locks the diskette in the current drive.
UnLock (O)	- This command write unlocks the diskette in the current drive.
Xfer (X)	- This command is much like the COPY command in the command processor except it does not do multiple files. You will be prompted for a source file and a destination file. Make sure to include the device names (i.e. 'D2:'). The screen is cleared before the copy. After its done, press the RETURN key to restore the display.
?Dir (V)	- This command displays the current directory path. To restore the display, press RETURN.
>Dir (T)	- This command displays the directory pointed to by the cursor. This is now the current directory.
<Dir (Y)	- This command displays the parent directory. This is now the current directory.
CreDir (N)	- This command creates a new subdirectory. Just enter the name of the new directory.
DelDir (D)	- This command deletes the directory pointed at by the cursor. No prompt is given (but the directory must be empty anyway before it may be deleted).

Other Notes About the MENU Program

Some of the commands invalidate user memory (destroy BASIC programs, etc.). They are as follows: **Copy, XInit, Load, Xfer,** and **Exec/P**.

While a cartridge is enabled, care must be taken to ensure that the command processor will not interfere with the MENU program. When you enter BASIC, a flag (called WARMFLG) indicates whether the contents of memory is valid. Both the command processor and the MENU program keep their own copies of this flag, but they may differ. There is no problem if **just** using the command processor **or** the MENU program, but there are some scenarios you **must** try to avoid when using both.

1. You load the menu ('MENU R'), enter BASIC (option 'RunCar'), write a BASIC program (or load one), type 'DOS' (you are now in the MENU), perform a memory destructive command (like 'Copy'), enter the command processor (option 'Exit'), and type 'CAR'. In this example, the command processor never knew that memory was destroyed.

2. You load the menu 'MENU R', enter BASIC (option 'RunCar'), write a BASIC program (or load one), type 'DOS' (you are now in the MENU), enter the command processor (option 'Exit'), perform a memory destructive command (like 'COPY'), and press RESET. In this example, the MENU program did not realize that memory was destroyed and RESET used MENU's copy of WARMFLG.

Important: If you enter a cartridge through a command, the program that you entered from will update WARMFLG. If you enter because of a RESET, MENU will update WARMFLG.

CHAPTER 12 — TIME AND DATE SUPPORT

Have you ever found yourself frustrated because you're not sure which file is the latest version of a program you wrote the day before (or even last year)? Well, for those who answered yes, SpartaDOS offers a solution to this problem. All versions of SpartaDOS support file time/date stamping, which allows you to know exactly when you saved a particular file. You need to know some more commands in order to, display the current time and date, to set a new time and date, and to update the time and date in a file. In this chapter, all the commands necessary shall be given.

To Activate Time/Date Clock

SpartaDOS has a few memory locations dedicated to holding the current time and date. When you boot the system, these locations contain a default time of 3:59:00pm and a date of 1/1/84. Unless something is done to change this, all files that you save will be tagged with this default value. (Try saving a file from BASIC before you install the clock.) There are two types of clocks available with SpartaDOS; one is the R-Time 8 cartridge which does not need to be set (unless for Daylight Savings etc.), the other type is a software clock that needs to be set every time you boot the system. To install a clock simply type TD (for R-Time 8 cartridge), or TIME. This will link a small program into the Atari that keeps the current time and date displayed. The display actually adds an extra line at the top of the screen, which will remain even in BASIC, as long as the BASIC programs do not modify the deferred VBLANK vector used. The clock also updates the memory locations within SpartaDOS so that when you save a file, the time and date displayed will appear in the directory along with the new file. Note: TIME, TD, and XTD are all relocatable, which means they load in above MEMLO, then move MEMLO just above their program area.

To Set Time/Date Clock

Normally you will not need to set the R-Time 8 clock, since it keeps time while the computer is off. But, if you don't have the R-Time 8 cartridge, you will need to set the software clock (TIME) every time the system is booted. TIME is a simple counter that uses the vertical blank interrupt to keep time. Once the correct time has been set (by the SET command), it will continue to operate like the R-Time cartridge (TD) until the computer is turned off. For instructions on how to set the R-Time cartridge or the software clock, refer to the SET and TSET command descriptions the follow in this chapter.

Note: The R-Time 8 Cartridge is a very accurate, crystal based timing device which works on both 60 Hz and 50 Hz systems. The software clock installed with the TIME command is not very accurate and will usually lose about 1 minute each day.

TIME Command

Purpose

To display the time and date at the top of the screen, and to install the time function into DOS. The X parameter turns the time and date display off. Similar to the TD Command but for use without the R-TIME 8 Cartridge.

Syntax

TIME [X]

Type and Restrictions

External under CP versions 1.x and 2.x

Remarks

If only TIME is entered, the time/date line will appear with the current time according to DOS. If TIME is already on, then nothing happens. If the 'X' parameter is entered, the time and date display is turned off but the clock stays installed. To change the time and date see the SET Command. To access this clock in your BASIC programs see Appendix D.

NOTE: This command patches itself in the initialization vector and is re-initialized with every RESET. The time/date routine stays in memory and moves MEMLO up. IF 'TIME X' is entered, the display is turned off but the module still resides in memory.

NOTE: TIME patches itself into the deferred VBLANK vector. During disk I/O the time will move sluggishly since it is doing CRITICAL I/O, but the time will quickly catch up when the disk operations are done.

SET Command

Purpose

This command allows the user to set the time and date after installing the clock with the TIME command.

Syntax
SET [mm/dd/yy] [hh/mm/ss]

Type and Restrictions
External under CP versions 1.x and 2.x

Remarks
If no parameters are specified, then the program will ask for the time and date, otherwise, the time and date specified on the command line will be used. If using our R-Time 8 cartridge with battery backup, the TSET command must be used.

Example

```
    SET
```

Since no parameters were specified, the prompt showing the current date and asking for a new date appears. Type in the new date using slashes as delimiters (5/12/84). When asked to enter the time, repeat the above steps using 24 hour time (13/01 results in 1:01:xxpm , 1 results in 1:xx:xxam).
NOTE: 'xx' in time/date indicates the standard default that was in the number location before SET.

```
    SET 12/10/84 21/12
```

This command line sets the date at 12/10/84 and the time to 9:12:xxpm. Notice that we use slashes as delimiters in the command line. Do **not** ever use colons in a SET or TSET command line or unpredictable things will happen!

TD Command

Purpose
Used with R-TIME 8 Cartridge to install the hardware clock and display the time and date on the first line. The 'X' parameter will turn the time and date display off but keep the clock installed.

Syntax
TD [X]

Type and Restrictions
External under CP versions 1.x and 2.x

Remarks
If TD is entered and the R-TIME Cartridge is present in the right (Atari 800 only) or left slot, the current time and date will appear on the top display line. TD uses the interrupt vectors to read the R-TIME Cartridge 60 times a second and update the display every second. If the R-TIME Cartridge is not installed then an error message is displayed. If the 'X' parameter is entered, the time and date display is turned off. To change the time or date in the R-TIME 8 Cartridge use the TSET Command.

The R-TIME 8 Cartridge is our real time clock calendar cartridge with battery backup. It can be used in either cartridge slot with any 8 bit Atari Computer including the new XE line. When using batch files with the TD command, it will boot up with the correct time and date without operator input. No cartridge memory is used by this device so it can be left in the slot even when not used. The R-TIME 8 Cartridge has an extension socket in the top so you can use it with another cartridge in the XL/XE computers.

NOTE: The TD command patches itself in the initialization vector and is re-initialized with every RESET. The time/date routine stays in memory and moves MEMLO up. If 'TD X' is entered, the display is turned off but the module still resides in memory.

NOTE: TD patches itself into the deferred VBLANK vector. During disk I/O the time will move sluggishly since it is doing CRITICAL I/O, but the time will catch up when the disk operations are done.

XTD Command

Purpose
Used with R-TIME 8 Cartridge to load the time and date into the system when you do not what the time/date display.

Syntax
XTD

Type and Restrictions
External under CP versions 1.x and 2.x

Remarks

If XTD is entered and the R-TIME 8 Cartridge is present in the right or left slot, the current time and date will be loaded into the system but not displayed. This command uses less memory than TD and is used with programs which don't like TD displayed on the top line. As with TD, XTD uses the interrupt vectors to read the R-TIME 8 Cartridge 60 times a second and update the display every second. If the R-TIME 8 Cartridge is not installed or not working then an error message will be displayed. To change the time or date in the R-TIME 8 Cartridge use the TSET Command.

NOTE: This command patches itself in the initialization vector and is re-initialized with every RESET. The time/date routine stays in memory and moves MEMLO up.

TSET Command

Purpose

This command allows the user to set the time and date in the R-TIME 8 Cartridge (See TD command).

Format

TSET [mm/dd/yy] [hh/mm/ss]

Type and Restrictions

External under CP versions 1.x and 2.x

Remarks

If no parameters are specified, then the program will ask for the time and date, otherwise, the time and date specified on the command line will be used. Be sure to install the clock cartridge first with either TD or XTD.

Example

```
TSET
```

Since no parameters were specified, the prompt showing the current date and asking for a new date appears. Type in the new date using slashes as delimiters (5/12/84). When asked to enter the time, repeat the above steps using 24 hour time (13/01 results in 1:01:xxpm , 1 results in 1:xx:xxam).

NOTE: 'xx' in time/date indicates the standard default that was in the number location before SET.

```
    TSET 12/10/84 21/12
```

This command line sets the R-TIME 8 Cartridge date to 12/10/84 and the time to 9:12:xxpm. Notice we use slashes as delimiters in the command line. Do **not** ever use colons in a SET or TSET command line or unpredictable things will happen!

CHTD Command

Purpose
This utility command is used to change a file's time/date stamp.

Syntax
CHTD [Dn:][path>]fname[.ext]

Type and Restrictions
External under CP versions 1.x and 2.x

Remarks
CHTD is used to change or correct the time/date stamp on a file. This can be useful for files which come from a DOS disk which doesn't support time/date stamping or when using a program which won't allow one of the clocks to be installed. This command takes the current system clock time and date (changed with SET or TSET, installed with TIME, TD or XTD) and writes it to the files which match the filespec. Wild cards are supported.

Example

```
    CHTD D:*.*
```

This takes the current time/date and writes it to all files within the current directory on the default disk.

CHAPTER 13 — COMMUNICATIONS SUPPORT

SpartaDOS supports several RS232 interfaces; included in SpartaDOS are the handlers for the ATR8000 serial port and the Atari 850 interface. Though handlers for other serial interfaces (e.g. R:Link) are not included in SpartaDOS, they should still work if used from SpartaDOS (as long as they relocate properly). With any of these interfaces along with the correct handler and MODEM program, you may dial up Bulletin Board Systems (BBS), or use a direct interface to other computers, etc.

MODEM or Terminal Programs

The RS232 device handlers link themselves into the system much like any Atari DOS does; but once it has linked itself in, there is no indication that anything has happened — of course something has, or it wouldn't exist. The point is, a terminal program (or MODEM program) is also needed to communicate to external devices. A MODEM program is basically a program that concurrently copies characters from K: to R: (what you type gets send out the RS232 line), from R: to E: (what comes in through the RS232 line gets displayed), and optionally echo K: to E: (Echo keystrokes to your screen) or echo R: to R: (FULL DUPLEX mode).

Note: K:, E:, and R: are device identifiers on the Atari for the KEYBOARD, the SCREEN EDITOR, and the RS232 device respectively.

Communicating Through Phone Lines

A MODEM is required if you want to communicate with another computer (or a BBS) through a phone line. In this case the MODEM translates the RS232 signals (i.e. from the ATR8000, 850 interface, etc.) into sound that passes through the phone lines and vice versa. The computer (or BBS) at the other end of the phone line has a similar set up. Its MODEM converts sound back into RS232 signals, thus two modems at both ends of a phone line act as though you ran a cable from your interface to the other computer's interface.

Two Modes of RS232 Handler Operation

Most Atari RS232 handlers operate in two modes. The simplest is **Block Mode** which stores data until a buffer fills. This results in normal SIO (Serial Input and Output) operation, which means data is sent and received much in the same manner as in the disk interface. The problem with this method is that you can't send and receive at the same time. A solution to this is called **Concurrent I/O**, which directly links the RS232 lines to the Atari SIO lines. The Atari is interrupted when a character has been received and places it in a buffer. When the Atari is ready to send a character, it immediately sends it through the SIO line. Block Mode is rarely used on the Atari since it does not allow smooth operation (responses can be disjointed), but Concurrent Mode also has a major drawback; operations that use the SIO can't be performed while in Concurrent Mode. MODEM programs solve this problem by switching in and out of Concurrent Mode when doing disk drive access. The ATR8000 handler also solves this problem by doing the same thing, but the switching is invisible to even the MODEM program. Who cares, right? Well, for those who have ATR8000s, try this:

```
AT_RS232
PRINT R:
-R:                    (Note: Make sure remote is at 300 baud)
```

This sequence allows a remote device to take control of your Atari! Of course your MODEMS must first be communicating, but the remote device actually has access to your disk files (and can type commands just as you would). There are a few hitches with this that make it unpractical (e.g. BASIC resets the Audio Registers if entered, there is no remote RESET or BREAK ability, and direct screen access programs won't work, etc.). One thing that might be useful is sending ASCII disk files between computers without any special MODEM program.

RS232 Commands

Purpose
To load the RS232 Handler for communications.

Syntax
RS232 or
AT_RS232

Type and Restrictions
External under CP versions 1.x and 2.x

Remarks

The RS232 command is used with the Atari 850 interface module to boot the RS232 Handler. This command can be used as part of a batch file for automatic loading. Unlike Atari DOS 2 (without MEM.SAV), you can go to BASIC then SpartaDOS and back to BASIC without rebooting RS232.

AT_RS232 is the handler for the ATR8000. No 850 interface is needed with it. AT_RS232 is a concurrent only handler though it is intelligent in that it enables and disables concurrent mode automatically as needed for disk access.

Example

```
    RS232
```

With the 850 Module connected properly and powered up, you will hear the familiar beep over your monitor or TV speaker which tells you the handler was successfully booted.

PORT Command

Purpose

To set speed, word size, stop bits, translation, input and output parity, and EOL parameters for RS232 communications.

Syntax

PORT [path>]fname[.ext]

Type and Restrictions

External under CP version 1.x and 2.x

Remarks

PORT sends a two byte configuration file to the RS232 port. The first byte is for XIO 36, Aux1 and the second byte is for XIO 38, Aux1. The first byte will set baud rate, word size, and number of stop bits to transmit. The second byte will set parity checking on input and parity on output, translation mode, and allow LF after CR.

These configuration files can be created with the COPY command but first you must figure the Aux 1 code by adding the values in the following tables. Then find the corresponding ATASCII keyboard code and create the desired two byte file with the 'COPY K: D:fname' command. (See COPY command for more detail).

To calculate XIO 36, Aux 1 — If you want:

110 baud	add 5	8 bit word	add 0
300 baud	add 0	7 bit word	add 16
1200 baud	add 10	6 bit word	add 32
1800 baud	add 11	5 bit word	add 48
2400 baud	add 12		
4800 baud	add 13	1 stop bit	add 0
9600 baud	add 14	2 stop bits	add 128

To calculate XIO 38, Aux 1 — If you want:

Translation:		End of Line (EOL):	
Light	add 0	No LF append	add 0
Heavy	add 16	Append LF	add 64
None	add 32		

For Input Parity Check:		For Output Parity Set To:	
None	add 0	None	add 0
Odd	add 4	Odd	add 1
Even	add 8	Even	add 2
Ignore	add 12	Mark	add 3

Example

```
PORT P__4800.RC
```

This will send the configuration file P__4800.RC to the RS232 port. P__4800.RC is an example of a configuration file which is included on the Master diskette. Its two bytes set the port at 4800 baud, no parity, 8 data bits, 1 stop bit, and send LF after CR on output.

CHAPTER 14 — INPUT/OUTPUT REDIRECTION

Input Redirection

Input redirection is the ability for a file (or device) to supply the input to your computer **as if you were typing it yourself.** This means that a file could "enter" commands you normally type for a certain process. Unlike other Atari DOS's, **all** input (not just commands for the command processor) is redirected. This is accomplished by using **Batch Files.**

Batch Files

Purpose

To retrieve and execute a file (**fname.BAT**) which instructs DOS to go perform specific operations in a specific order. **STARTUP.BAT** is a special batch file which is automatically executed when the diskette is booted.

Syntax

-fname

Type and Restrictions

Internal under CP versions 1.x and 2.x

Remarks

A batch file contains executable DOS instructions. It can be created with a word processing program or with the Screen Editor using the COPY command. You can use the TYPE command to view the contents of a batch file. A typical example of a batch file could be to load an RS232 handler, go to the BASIC cartridge, and then RUN a communications program. All batch files **must** end with the filename extension of .BAT (except with versions 2.x). Comments can be added to your batch files by typing a semicolon (;) at the beginning of the command line. The maximum length of a command line is 64 characters. Each command line is terminated by pressing the RETURN key. To execute a batch file type a dash (-) then the filename and RETURN. (Do **not** type the extension unless using CP version 2.x and then only if the batch file does not end in '.BAT'.) Pressing SYSTEM RESET while a batch file is running aborts the batch operation and goes directly to DOS or the cartridge if present.

Generally a disk will have a STARTUP.BAT file which will initialize things the way you want them for that particular diskette. For instance you may want the R-TIME 8 cartridge to be installed if it is present, the keyboard buffer installed, and then the screen cleared. To create that batch file you could use the following keystrokes:

```
COPY E: D:STARTUP.BAT
TD
KEY
;<ESC><CTRL + CLEAR>
<CTRL + 3>
```

In the above example, <ESC> means to press the ESC key, and <CTRL + CLEAR> means to press the CTRL key and hold it down while you press the CLEAR key.

There are some special command files for use in batch files. PAUSE.COM (CP version 1.x) will stop execution of the batch file until another key is pressed, and DIS_BAT.COM (CP version 1.x) will disable batch file processing. PAUSE is internal with CP version 2.x and XDIV is the internal command to disable batch files with CP version 2.x.

NOTE: While the command is in effect, IOCB #5 may **not** be used, since this is the IOCB the input goes through. This IOCB acts as if it were closed, meaning that it could be opened (this **will** have bad side effects on the system and cause unpredictable results). The reason for making the IOCB appear closed, is to prevent the system from closing the file; e.g. BASIC when entered, closes all IOCBs.

SPCOPY, FORMAT, INIT, and other commands, may re-initialize the DOS which will terminate batch execution. Batch files **can not** be used to call up other batch files (linking) with CP version 1.x. CP version 2.x does allow linking of batch files (i.e. the last command in a batch file can be '-fname').

Example

```
-MODEM
```

This command will execute the set of instructions saved under the filename MODEM.BAT on the default drive under the current directory. This file might look like the following example:

```
RS232
CAR
RUN "D:AMODEM4"
```

This batch file when executed will run a file called RS232.COM (link in the RS232 handler), go to the BASIC cartridge, and then RUN the BASIC program AMODEM4.

PAUSE Command

Purpose
To temporarily halt execution of a batch file and to prompt the user for a response to continue.

Syntax
PAUSE

Type and Restrictions
External under CP version 1.x
Internal under CP version 2.x

Remarks
This is a convenient way to stop the screen while displaying instructions from a batch file. **Caution:** when using PAUSE with SpartaDOS, do not swap disks during a PAUSE command as this may destroy the second disk! Abort the PAUSE with a SYSTEM RESET.

Example
Consider execution of the following batch file from drive 1:

```
RS232
; Please insert your communications
; program diskette into drive #2
;
PAUSE
CAR
RUN "D2:AMODEM4.2"
```

This batch file will first load the RS232 Handler from the 850 interface then display the next 3 comment lines and stop with the display 'Press any key to continue'. After the user follows the instructions and presses a key, this program will go into the BASIC cartridge and run the modem program specified.

TYPE Command

Purpose
To display the contents of an ASCII file. Commonly used to read a batch file without executing it.

Syntax
TYPE [Dn:][path>]fname[.ext]

Type and Restrictions
Internal under CP versions 1.x and 2.x

Remarks
The file is read, line by line, and printed to the screen editor. If a line is longer than 64 characters, an error will occur (truncated record). This command will only print one file. This same function could also be done with the COPY command, however the COPY command will erase the contents of program memory. Use TYPE with the PRINT Command (redirect output) to 'type' a file to the printer.

Example
```
    TYPE STARTUP.BAT
```

This command displays the contents of the batch file used for initialization.

Output Redirection

Output redirection is the ability to echo everything that gets written on the screen, to another output device (or disk file). This means that you can get a hardcopy of everything that transpires on the computer. The only exceptions (data that will not be echoed) are programs that write directly to the screen (like MENU.COM and most binary games). The PRINT command sets the device (or file) that output is to be echoed to.

PRINT Command

Purpose
To echo all output that is written to the screen editor (E: through IOCB #0) to a specified output device.

Syntax
PRINT [dn:][path>]fname[.ext][/A] or
PRINT d[n]: or
PRINT

Type and Restrictions
Internal under CP versions 1.x and 2.x
The '/A' option is only allowed under CP version 2.x

Remarks
This command is normally used to send everything that gets printed on the screen to the printer. However, the output may go anywhere the user desires, including a disk file. This feature is very useful if one wants to have the output of a BASIC program, or an editing session, etc., go to the printer.

In CP version 1.x, the PRINT command acts like a toggle; the first time the output goes to the device/file specified; the second time, the command closes the file and output returns to normal.

In CP version 2.x, the PRINT command can be chained. This means that the last PRINT's file/device is closed, and output is echoed to the new file/device. If no parameter is given after the PRINT command, the file/device is closed and it does **not** open another. The '/A' option allows the user to append the output to the end of an existing file.

NOTE: While the command is in effect, IOCB #4 may **NOT** be used, since this is the IOCB the output goes through. This IOCB acts as if it were closed, meaning that it could be opened (this **will** have bad side effects on the system and cause unpredictable results). The reason for making the IOCB appear closed, is to prevent the system from closing the file; e.g. BASIC when entered, closes all IOCB's.

Example
```
    PRINT P:
```

This command sends all future screen display to the printer until PRINT toggles this off.

```
    PRINT D1:SAVIT.NOW
```

Sends all future screen display to a file on drive #1 called SAVIT.NOW until PRINT is entered.

How the I/O Redirection Works

On the Atari computers, I/O redirection is possible because of something called the **device table**. This is a list of device letters (e.g. E K S P D. . .) followed by a **handler table** address. SpartaDOS patches itself into the device table and saves a pointer to its own handler table. The handler table is a list of pointers to routines such as Get character, Put character, Open file, Close file, Status, and XIO. The SpartaDOS handlers then check that the IOCB in use is #0. If so, the DOS will take appropriate action to input from or output to the E: (Editor) device.

Disabling I/O Redirection

Sometimes the I/O redirection can get the system into trouble. The only time it really happens is when trying to run a binary game (saved as a DOS file). This is because a few games (not all!) tend to move themselves on top of DOS, and then output data through the E: device (crash. . . SpartaDOS is no longer there to handle the Editor (E:) output). Thus, there is a need to restore the operating system's (OS) handler printer in the device table. The following commands perform this function.

DIS__BAT Command

Purpose

The DIS__BAT command is used with CP version 1.x to disable batch processing and the PRINT command (redirection of I/O). This may be necessary in order to run certain programs. If using CP version 2.x see the XDIV command.

Syntax

DIS__BAT

Type and Restrictions

External under CP versions 1.x and 2.x
XDIV is preferable under CP version 2.x

Remarks

Certain programs will not run under SpartaDOS unless the batch file processing is removed. DIS__BAT will disable this and allow most of those programs to run correctly. DIS__BAT can be run as the last command of a batch file. SYSTEM RESET re-enables I/O redirection **if disabled with DIS__BAT.**

XDIV Command

Purpose
The XDIV command is used with CP version 2.x to disable batch processing and the PRINT command (redirection of I/O). This may be necessary in order to run certain programs. **If using CP version 1.x see DIS_BAT.**

Syntax
XDIV

Type and Restrictions
Internal under CP version 2.x

Remarks
Certain programs will not run under SpartaDOS unless the I/O redirection is removed. XDIV will disable this and allow most of those programs to run correctly. XDIV can be run as the last command of a batch file. Unlike the DIS_BAT command, XDIV is permanent until the computer is rebooted.

CHAPTER 15 — KEYBOARD BUFFERS

The Need For a Keyboard Buffer

Do you ever find yourself trying to type ahead of the computer? For example, you want to load the RS232 handler, go into BASIC, and then run a MODEM program. No matter how fast the computer is, you still want to type 'CAR' before the computer has even had time to figure out the 'RS232' you just typed. Well, again SpartaDOS offers a solution to this 'urge.' Two commands are available (KEY for non-XL/XE's, XKEY for XL/XE's) that give you a 'buffer' for keystrokes made ahead of the computer. As an added feature, the key repeat rate as been increased to double that of normal. The commands follow.

KEY & XKEY Commands

Purpose

To install a 32 character keyboard buffer.

Syntax

KEY or
XKEY

Type and Restrictions

External and CP versions 1.x and 2.x
XKEY works on XL/XE computers
KEY works on non-XL/XE computers

Remarks

One of the things we always miss after working on a larger computer is a large keyboard buffer. The standard Atari 8-bit computer has a one character buffer, but the KEY/XKEY keyboard buffers allow type ahead while the computer is tied up with other functions (e.g. disk I/O, printing, etc.). After executing this command, you will have a 32 character keyboard buffer that is functional even while in BASIC!

Example

```
    KEY
```

The buffer is now installed. You can now do a 'TD', 'DIR', 'RS232', 'CAR', and 'RUN "D:MODEM" ', without waiting for the computer to catch up!

CHAPTER 16 — INFORMATIONAL COMMANDS

Memory Related Commands

The following are three commands relating to the allocation of memory. The **BUFS** command sets the number of buffers for CP version 1.x (which has a direct bearing on how fast the DOS performs). CP version 2.x has no BUFS command since it maintains 12 buffers at all times (which are underneath the OS ROM's). The **MEMLO** and **MEM** commands simply display the lower and upper bounds of memory (MEMLO only displays lower bound). They are really only important if you are doing machine language programming. The command descriptions follow.

MEMLO and MEM Commands

Purpose

To display MEMLO (lower bound) and MEMHI (upper bound — **only MEM displays this**).

Syntax

MEM or
MEMLO

Type and Restrictions

MEM is internal under CP version 2.x
MEMLO is external under CP versions 2.x and 1.x

Remarks

The MEM command displays MEMLO and MEMHI in hexadecimal notation. These values can be useful since many of our files are relocatable and can move MEMLO up in memory. With this command you can see just how much memory one of these relocatable files takes. MEM will also give you an idea of the free memory available. You can see if the internal BASIC (XL/XE) is installed by noting the MEMHI location. The MEMLO command only displays MEMLO.

Example

```
MEM
```

This command displays the MEMLO and MEMHI values in the format:

```
Memlo = $xxxx Memhi = $xxxx
```

Where 'xxxx' is any hexadecimal number.

BUFS Command

Purpose
To set or check the number of buffers currently in use under CP version 1.x only.

Syntax
BUFS [n]

Type and Restrictions
Internal under CP version 1.x

Remarks
BUFS will display the number of 128 byte blocks of memory (buffers) currently reserved for DOS use. This display is a DECIMAL value between 2 and 16. 'BUFS n' will set the number of buffers to be reserved for DOS use, where 'n' is a **hexadecimal** (HEX) value between $2 and $10 (the '$' means HEX — $10 is 16 in decimal). The boot up default is 4 under STANDARD.DOS, and 6 under the other version 1.x DOS's. This default can be changed when formatting a new version 1.x diskette by using the **INIT** command.

NOTE: More buffers require more memory, which moves MEMLO up (the lower memory boundary). The minimum requirement for single density read and write is 2, and for double density read and write is 4. In general, if the program requires reading and writing in random fashion, the more buffers you have, the faster the operation will be, and the less wear on your disk drive.

SpartaDOS version 2.x has 12 buffers built in so the BUFS command is **not** needed.

Example
```
    BUFS F
```

The above command sets the number of DOS buffers to decimal 15.

```
    BUFS
```

This command results in the output of 'BUFS = n' where 'n' is the current number of buffers in use (in decimal).

Disk Drive Related Commands

Two commands are included which allow you to check your drive speed, and to determine the amount of free space on your diskettes. The number of 'free bytes' and 'total bytes' are an estimate based on the number of 'free sectors' and 'total sectors' respectively. The commands follow.

CHKDSK Command

Purpose

To display the volume name, random & sequence numbers (version 2.x diskettes only), sector size, formatted bytes on disk, available bytes on disk, and write lock status (version 2.x diskettes only).

Syntax

CHKDSK [Dn:]

Type and Restrictions

Internal under CP version 2.x

Remarks

This command is used for determining information about a diskette. If the diskette was initialized with XINIT (version 2 format), CHKDSK will give a volume name then random & sequence number. SpartaDOS 2.x uses a random number (created when formatted) and a sequence number (which increments when a file is opened for write) to identify the disk (in addition to the volume name). The next line is the 'bytes/sector,' which will be either 128 or 256 depending on the format. The next line is the 'total bytes,' which is the total formatted capacity before any data is written. The 'Bytes free' is the amount of available space left on the diskette. The 'Write lock' is a type of the software write protect ('ON' indicates locked or protected).

If the diskette is **not** a version 2 format, the random & sequence numbers and write lock status are omitted from the display. If the diskette is an Atari DOS 2 type format then these plus the volume name are omitted from the display.

Example

```
    CHKDSK
```

A hypothetical double density, single sided, SpartaDOS 2.x disk might display:

```
      Volume:  Games1 0A 25
Bytes/sector:  256
 Total bytes:  184320
  Bytes free:  123390
  Write lock:  ON
```

RPM Command

Purpose
To display the drive speed in RPM for user information.

Syntax
RPM [Dn:]

Type and Restrictions
External under CP versions 1.x and 2.x

Remarks
This command will start the drive spinning and read a sector continuously while updating the display every second with the actual speed your drive is turning in RPM (revolutions per minute). The correct speed is 288 and can be adjusted by turning VR2 on a 1050 drive or R104 on an 810 drive. **Note:** a formatted disk must be in the drive under test with the drive door closed. Pressing any key will stop the RPM test.

Example

```
RPM D2:
```

The display will show 'Drive RPM is XXX', where XXX is the actual speed of drive 2.

CHAPTER 17 — MACHINE LANGUAGE SUPPORT

Loading, Saving, and Running

SpartaDOS provides several commands to allow you to load, save, and run binary files (machine language). These tend to be for more experienced programmers but it doesn't hurt to understand them also. All the '.COM' files on the distribution diskette are called 'command files.' Generally, when you write your own utilities, you will want to make them into 'commands' also. Chapter 19 should be read to understand how to interface with the command line (so that parameters may be used).

Command Files

Purpose

To load and run binary files. Also, it also provides a standard for you to pass parameters to machine language programs.

Syntax

[Dn:][path>]fname [parameters]

Type and Restrictions

Supported by CP versions 1.x and 2.x
(By definition of a CP — 'command processor')

Remarks

Command files are executable binary (machine language) files. If you try to execute a non-binary file you will get an error 152 or a 'Not binary file' message with CP version 2.x. All binary files begin with an $FF $FF header. With CP version 1.x all command filenames must use the extension '.COM'. To load and run these files, type the 'fname' portion of the filename only (not the '.COM'). Wild cards are supported with both CP versions.

CP version 2.x has the added capability of treating any binary file as a command file. '.COM' is the assumed default extension. To load and run a file with any other extension, type the full filename including the extension. If there is no extension on the filename, be sure to end the command with a period (.).

Command files are run at the beginning of the first segment after they are loaded. RUN and INIT addresses ($2E0 and $2E2) are also supported with the RUN address ($2E0) having priority over the run-at-beginning-of-first-segment address.

Example

```
    TYPING
```

The above example will load and run a file under any SpartaDOS called 'TYPING.COM'.

```
    TYPING.
```

This will load and run the binary file called 'TYPING' under CP version 2.x.

LOAD Command

Purpose

This command loads any binary file into memory and does not run the file. The standard DOS RUN and INIT vectors are **not** used.

Syntax

LOAD [Dn:][path>]fname[.ext]

Type and Restrictions

Internal under CP versions 1.x and 2.x

Remarks

This command is useful for loading character sets, binary data, or files that should not be run. Note that a load can only be done from the 'D' device, since load is now an XIO function of the 'D' handler.

Example

```
    LOAD MYFILE.OBJ
```

This loads a file called MYFILE.OBJ into the memory locations specified in its header(s) but does not RUN the file.

NOTE: Don't get this confused with the LOAD command in BASIC. This LOAD is similar to the BASIC command but it loads only binary files (with a header of $FF $FF). BASIC programs are relocatable while many binary files are not and can cause system crashes (if they load over DOS or other volatile areas).

RUN Command

Purpose
To re-execute the last '.COM' file or execute at a given address. (To load and run a binary file see 'Command Files' or the 'MENU' section.)

Syntax
RUN [address]

Type and Restrictions
Internal under CP versions 1.x and 2.x

Remarks
If an address is not specified, then the last .COM file is executed. RUNLOC contains the address of the last command (see technical notes). If you specify an address, execution begins there, and RUNLOC will be updated with that address. Note that 'address' is in HEX notation.

Example
```
RUN 4000
```

This command starts executing a program at memory location $4000.

```
RUN
```

This command runs the last file executed. If SPCOPY was run and you use the OPTION key to get back to DOS, then typing RUN will take you back into SPCOPY as long as the file was not destroyed in memory. (This can be a great time saving feature).

NOTE: Don't get this confused with the RUN Command from BASIC. This RUN command is meant to RUN binary (machine language) files, not tokenized BASIC programs.

SAVE Command

Purpose
This command saves binary data from memory to disk. To append data, see the APPEND command, or with CP version 2.x use the '/A' option.

Syntax
SAVE [Dn:][path>]fname[.ext][/A] address address

Type and Restrictions
Internal under CP versions 1.x and 2.x
The '/A' option is only allowed under CP version 2.x

Remarks
This command saves a block of data with the first address being the start memory address and the second address being the ending memory address. The file is saved in the same format as all binary files on the Atari. An $FF $FF header is written first, followed by the start/end address, and then the data. Remember that 'address' is a number in HEX notation.

When using CP version 2.x, the '/A' option allows the SAVE command to work exactly like the APPEND command; appending the block of data onto the existing file specified (except APPEND does **not** write the $FF $FF header).

Example
```
    SAVE D1:CODE.OBJ 8000 9FFF
```

This command saves the memory from $8000 to $9FFF in a file called CODE.OBJ.

APPEND Command

Purpose
This command saves a binary block of data at the end of an existing binary file.

Syntax
APPEND [Dn:][path>]fname[.ext] address address

Type and Restrictions
Internal under CP versions 1.x and 2.x

Remarks
The format is the same as in the SAVE command. The file specified should already exist since the $FF $FF header is **not** written. Also the file is opened for append/write rather than just write as in the SAVE command. Remember that 'address' is in HEX notation. APPEND can also be accomplished with CP versions 2.x by using the '/A' option with the COPY, SAVE, or PRINT commands. Do **not** try to use the '/A' option with the APPEND command. Garbage will result.

Example

```
APPEND D1:GAMES>GHOST.COM 4000 47FF
```

The above command appends the block of memory from $4000 to $47FF onto the end of the file called 'GHOST.COM' on the disk in drive #1 under the existing subdirectory called 'GAMES'. This is a command primarily for advanced users working in assembly language.

Informational Commands

The next three commands are for the more experienced programmers. The DUMP and MDUMP commands give straight HEX dumps of a file and memory respectively. The OFF__LOAD command is a relocating load command used for loading programs at locations other than their native load addresses. Descriptions of these commands follow.

DUMP Command

Purpose

This utility will display a file or portion of a file in HEX and ATASCII or ASCII format.

Syntax

DUMP [Dn:][path>]fname[.ext] [start [#bytes]] [/P]

Type and Restrictions

External under CP versions 1.x and 2.x
The optional 'start' and '#bytes' parameters are not allowed when using an Atari DOS 2 formatted diskette.

Remarks

The DUMP command is used to find valuable information about a file. The 'start' parameter is the beginning offset (HEX) in the file that you want to start dumping from (default is 0). If you try to point past the end of the file you will get an error message 'Address range error'. The '#bytes' parameter is the number of bytes (HEX) you would like to have displayed. The screen will show the file position (in HEX) at the left, the values of eight memory locations across the center, and the ATASCII representation at the far right. The optional '/P' parameter will replace the control characters with periods (.) leaving only ASCII text at the far right. This is useful if you want to redirect the output of DUMP to a printer v the PRINT command.

Example

```
    DUMP TEST.OBJ 1000 5 /P
```

This command displays the hex values at file positions $1000 through $1004 of the file TEST.OBJ and also displays the ASCII equivalents while substituting any control characters with periods (.).

MDUMP Command

Purpose
This utility will display memory locations in HEX and ATASCII or ASCII format. It is very similar to DUMP but works on blocks of memory rather than files.

Syntax
MDUMP [address [#bytes]] [/P]

Type and Restrictions
External under CP versions 1.x and 2.x

Remarks
The MDUMP command is used to display the contents of specific memory locations. The screen will show file position (in HEX) at the left, the values (in HEX) of eight memory locations across the center, and the ATASCII representation at the far right. The optional '/P' parameter will replace the control characters with periods (.) leaving only ASCII text at the far right. This is useful if you want to redirect the output of MDUMP to a printer with the PRINT command.

Example

```
    MDUMP 2E0 2
```

This command displays the values (in HEX) of bytes $2E0 and $2E1 along with their ATASCII equivalents.

OFF_LOAD Command

Purpose
This utility command loads in files at an offset and optionally displays segment addresses, file position for beginning of segment, and can query whether to load a given segment. If may also be used to create non-relocatable versions of OFF_LOAD.

Syntax

OFF__LOAD [Dn:][path>]fname[.ext] offset [/SNPQ] or
OFF__LOAD -R address [Dn:][path>]fname[.ext]

Type and Restrictions

External under CP versions 1.x and 2.x
The 'N' and 'Q' parameters may not be used when 'OFF__LOADing' from an Atari DOS 2 formatted diskette.

Remarks

OFF__LOAD is a utility which is used to load segments of a file at given addresses. The offset is a number from 0 to $FFFF (in HEX). The 'S' parameter displays the start and end addresses of each segment and the new start address with offset. The 'N' parameter indicates that the segments are **not** to be loaded. This can be used with the other parameters to get address information without loading anything. The 'P' parameter causes the file position of that segment to be displayed. The 'Q' parameter (Query) stops before it loads each segment and asks 'Load this segment ?'. Answer with Y or N. The standard OFF__LOAD is relocatable and LOADs at $B400 then RUNs just above MEMLO. It intentionally will not function with a language cartridge installed.

The second OFF__LOAD format relocates the OFF__LOAD file to LOAD **and** RUN at 'address', and then writes it to the file specified by 'fname'. The purpose of this is to move the OFF__LOAD program out of the way of the cartridge area if necessary.

Example

```
    OFF__LOAD TEST.COM 0 /NS
```

The above command will display the segment addresses of the file TEST.COM but not load the file.

PUTRUN Command

Purpose

This command appends the RUN vector containing the start address of an external command file to the file. This is to make a command such as MENU, able to run as an AUTORUN.SYS (when only RUN/INIT vectors are used).

Syntax
PUTRUN [Dn:]fname[.ext]

Type and Restrictions
External under CP versions 1.x and 2.x

Remarks
This command is actually only useful for making command files into 'load and run' files for running under Atari DOS 2.

CHAPTER 18 — DISK DRIVE I/O

This chapter describes the way the disk drive handles its reading and writing of sectors, and a little about the SpartaDOS interface to the drive. Toward the end of the chapter, the VERIFY command (CP version 2.x only) is described in detail. The interface to US Doubler shall also be commented upon (and its high speed interface). Hopefully, after reading this chapter, any misconceptions you may have will be cleared up.

Basic Operation WITHIN the Disk Drive

All Atari disk drives are intelligent, meaning that the computer (800XL etc.) doesn't have to worry about talking directly to the surface of the diskette. Yet, it (the computer) must still be able to retrieve the information from the diskette. This is accomplished by a three way interface **within** the disk drive. These interfaces are as follows:

1) **The computer and disk drive interface** - This is commonly referred to as the SIO (serial input/output). All Atari devices (except the cassette) have a standard they abide by (which is discussed at great length in the Atari Technical Notes for the 800). Basically the SIO operation is as follows (step by step):
   a) The computer sets the COMMAND line low (to ground). This is one of the SIO port's lines.
   b) The computer sends a command frame. This consists of four bytes. They are 1) device ID — each unit on the serial port has a unique device ID, 2) command — such as read, write, status, and format, 3) two bytes of auxiliary information — such as the sector number high and low bytes. The command frame is followed by a checksum of the bytes of the command frame.
   c) The computer releases the COMMAND line by bringing it HIGH.
   d) The device (disk drive) identified by the device ID, answers by sending an ACK (if the command is valid) or a NACK (if the command is invalid).
   e) If the drive needs a data frame (as in the write sector command), the computer will send a data frame (the sector data) followed by a checksum.
   f) If (e) occurred and the data is good, the drive will send an ACK, if the data is bad a NACK is sent.

g) The disk drive performs the requested operation. When done, the drive sends either a COMPLETE or an ERROR code.
h) If doing a read type of operation, the drive will send the computer a data frame (the sector data) followed by a checksum.

2) **The drive CPU to controller interface** - There is a special chip in the disk drive called a controller. This device manages the specifics of the diskette format, does the seeks for sectors, and hand feeds the CPU the sector data. The CPU will simply supply the controller with the command, sector, and track numbers. The CPU sends data to the controller from its buffer (on a write) and receives data into its buffer (on a read).

3) **The drive CPU to drive hardware interface** - This last interface includes things like the drive motor, the stepper motor (moving track to track), the door and write protect sensors and various other controls.

SpartaDOS Buffer Management

SpartaDOS's sector buffer management is entirely different from the type used with Atari DOS 2. SpartaDOS dynamically allocates blocks of memory for sector buffering. This means SpartaDOS does not require a buffer for each drive to be used and does not need buffers for each open file. Theoretically, you may have 7 files open on 7 different drives using only 1 buffer in single density or 2 if double density (however, don't expect great speed).

Drive Access Vector

Did you ever wonder how the Ramdisks linked themselves into the system, or how AT_RS232 was able to switch concurrent I/O mode on and off? Well, this is done by providing a vector that **all Drive accesses through DOS** use. The Ramdisk simply checks the device ID and takes over if there is a match with its ID. Also, all SpartaDOS commands (e.g. DUPDSK, INIT, XINIT, etc.) use this vector so they can take advantage of the high speed I/O. For more information on this vector, refer to Chapter 19.

US Doubler — High Speed I/O

The US Doubler has two sets of serial routines; one being the standard set, and the other begin the high speed set. The routine that monitors the COMMAND line reads the command frame in one speed, and if an error (in checksum) occurs, the CPU switches modes and will try for a short period of time to receive in the new mode. Once speeds have been matched, that speed becomes the default. SpartaDOS, when it boots, does a '?' command (refer to Appendix G) to determine just how fast the US Doubler high speed I/O is. If that was successful, SpartaDOS continues to operate that drive at the high speed mode. Note: utilities by other software companies will normally use the $E459 SIO vector, so they will run at the normal speed.

Write With Verify

Many people seem to have misconceptions about write with verify. Verification occurs **within** the disk drive after it has written the sector. It is often believed that the computer does the verification by re-reading the sector. The reasons that SpartaDOS does not default to write with verify are because, 1) most drives are extremely reliable and, 2) it is three times slower than normal speed write without verify (even with the US Doubler). The reason it is so slow is because of the sector skew. Sectors are not in sequential order on Atari diskettes; they are optimized to allow two sequential sectors to either be read or written in one revolution. Thus 10 sectors may be read in 5 revolutions (which is about 1 second). With verify on, it takes 1.5 revolutions to write a sector (which is about 3 seconds to write 10 sectors). The US Doubler optimizes the skew so that it can read 3.5 sectors in single density and 3 sectors in double density in one revolution. (Normal double density drives can only read 1 sector per revolution, thus the 3X speed factor in double density.) If you are having trouble with your drive, you may want to have it serviced, or until then use verify. The VERIFY command description follows.

VERIFY Command

Purpose

This command changes the write mode to write with verify or write without verify.

Format

VERIFY ON or
VERIFY OFF

Type
Internal under CP version 2.x

Remarks
Verify means to write a sector, read it back, and compare it with memory before going on to the next sector. This takes much more time than just writing to disk, but it may be able to trap write errors. VERIFY ON sets the write mode to verify whenever writing; VERIFY OFF turns verify off. Most of us never use verify and have not had problems but it is nice to have it available, especially if you're having drive problems.

Note: when you format a version 1.x diskette, you may select if you want the DOS to verify by choosing to modify default parameters. If you select to verify, the DOS (when booted) will do all its writes with verify.

CHAPTER 19 — THE TECHNICAL STRUCTURE OF SPARTADOS

This chapter tends to be hard to grasp if you haven't been around computers much. It gives as many details of the DOS as it can. It starts out fairly easy with the BASIC XIO functions and then steps quickly into the machine language world.

SpartaDOS Functions From BASIC

The following is a list of SpartaDOS functions and how to implement them from BASIC through XIO statements. The DOS command, if applicable, follows the function name in parenthesis.

Note: throughout these examples, 'IOCB' represents an Input/Output Control Block number from 1 through 7. Atari Diskettes refers to Atari DOS 2 type formatted diskettes in either single or full double density.

Open a File

Syntax
OPEN #IOCB,T,X,"Dn:fname.ext"

Notes
This command opens a disk file through SpartaDOS. 'T' is the mode to open the file in (output, input, update, directory, etc.). The following are legal values of 'T' and what they do.

4 Open the file in read only mode.
6 Open a formatted directory. This returns a directory listing as in the DIR or DIRS command. 'X' indicates the style of directory. If 'X' is 128, then the directory is in the expanded format unless you reading an Atari DOS 2 diskette. If 'X' is 0, then a short directory format is given.
8 Open the file in write only mode. Note: any command that operates on an OPEN in write only mode (8), can use the '/A' option (CP version 2.x only) to force the OPEN to an OPEN in append mode (9).
9 Open file in append mode. Data is written to the end of an existing file.
12 Open for update mode. This mode allows you to read or write a file.

20 Open the current directory in read mode. This is the "raw" data of the directory and can be read as any other file. **This mode is only available on SpartaDOS formatted diskettes.**

24 Open the current directory in update mode. This is the "raw" data of the directory and can be read or written as like any other file. **This mode is only available on SpartaDOS formatted diskettes.**

36 Open a subdirectory in read mode, but the subdirectory is to be read as if it were a regular file. **This mode is only available on SpartaDOS formatted diskettes.**

You must be careful of the read and update of unformatted directory and subdirectory modes. If a mode like 40 (subdirectory + write) is used, you may destroy the entire subdirectory. The above listed modes are the only modes that are really useful to a user. Functions like CREDIR and DELDIR are the only routines that are have legitimate use for a mode like 40.

Rename File(s) (RENAME)

Syntax
XIO 32,#IOCB,0,0,"Dn:[path]fname.ext fname.ext"

Notes
The IOCB must be closed for this operation to be used. Wild cards may be used in the filenames. This function is valid for any format diskette. **Atari Diskettes may only be accessed with a version 2 SpartaDOS loaded into the computer!**

Erase File(s) (ERASE)

Syntax
XIO 33,#IOCB,0,0,"Dn:fname.ext"

Notes
The IOCB must be closed for this operation to be used. Wild cards may be used in the filename. This function is valid for any format diskette. **Atari Diskettes may only be accessed with a version 2 SpartaDOS loaded into the computer!**

Lock Diskette (LOCK)

Syntax

XIO 34,#IOCB,0,0,"Dn:"

Notes

The IOCB must be closed for this operation to be used. **This function is valid for only SpartaDOS version 2.x diskettes and must be used with a version 2.x SpartaDOS loaded into the computer!**

Protect File(s) (PROTECT)

Syntax

XIO 35,#IOCB,0,0,"Dn:[path>]fname[.ext]"

Notes

The IOCB must be closed for this operation to be used. Wild cards may be used in the filenames. This function is valid for any format diskette. **This function must be used with a version 2.x SpartaDOS loaded into the computer!**

Unprotect File(s) (UNPROTECT)

Syntax

XIO 36,#IOCB,0,0,"Dn:[path>]fname[.ext]"

Notes

The IOCB must be closed for this operation to be used. Wild cards may be used in the filenames. This function is valid for any format diskette. **This function must be used with a version 2.x SpartaDOS loaded into the computer!**

Set File Position - POINT

Syntax

X = POS
Y = 0
POINT #IOCB,X,Y **or**

```
Y = INT(POS/65536)
POKE 846+IOCB*16,Y
POS = POS-Y*65536
Y = INT(POS/256)
POKE 845+IOCB*16,Y
POKE 844+IOCB*16,POS-Y*256
XIO 37,#IOCB,0,0,"Dn:"
```
or

```
POINT #IOCB,SECTOR,OFFSET
```

Notes
For SpartaDOS Diskettes: In the first method, position (POS) **must** be from 0 to 32767. The second method may take positions up to 8,388,607 ($7FFFFF in HEX notation). You may position beyond the end of file if the file is opened in read/write mode. The space between the EOF and where you point is filled with zeros, but physically, no sectors are used to hold the zero data. Thus, it is possible to have a file 32K in length but only 5 sectors long. If the data in the gap is accessed in any way, a sector will be created for the 128 or 256 byte area around the location accessed.

Note: POINT under SpartaDOS uses an absolute position relative to the beginning of the file. This is different from the sector number and position byte as in Atari DOS 2.

For Atari DOS 2 Diskettes: In the third method, the POINT command gives a sector number and an offset within the sector. This is not a relative file position as in SpartaDOS formatted diskettes. This works identically like Atari DOS 2. **Atari Diskettes may only be accessed with a version 2 SpartaDOS loaded into the computer!**

Get Current File Position - NOTE

Syntax
```
NOTE #IOCB,X,Y
POS = X
```
or

```
XIO 38,#IOCB,0,0,"Dn:"
POS = PEEK(846+IOCB*16)*65536
POS = POS + PEEK(845+IOCB*16)*256
POS = POS + PEEK(844+IOCB*16)
```
or

```
NOTE #IOCB,SECTOR,OFFSET
```

Notes
For SpartaDOS Diskettes: In the first method, position (POS) **will** be from 0 to 32767. The second method will give positions up to 8,388,607 ($7FFFFF in HEX notation). Note that this is an absolute position relative to the beginning of the file. This is different from the sector number and position as in Atari DOS 2.

For Atari DOS 2 Diskettes: In the third method, the NOTE command gives a sector number and an offset within the sector. This is not a relative file position as in SpartaDOS formatted diskettes. This works the same as Atari DOS 2. **Atari Diskettes may only be accessed with a version 2 SpartaDOS loaded into the computer!**

Get File Length

Syntax

```
XIO 39,#IOCB,0,0,"Dn:"
POS = PEEK(846+IOCB*16)*65536
POS = POS + PEEK(845+IOCB*16)*256
POS = POS + PEEK(844+IOCB*16)
```

Notes
This returns the current file length (end-of-file pointer) of the currently open file. Note that this is **only works for SpartaDOS** formatted diskettes. Atari formatted diskettes have no equivalent.

Load Binary File (LOAD)

Syntax

```
XIO 40,#IOCB,4,X,"Dn:[path]fname.ext"
```

Notes
This command will load a binary file. If X is less than 128, then the INIT/RUN vectors will be used, otherwise they will be ignored. Note that the IOCB must not be open. **Atari Diskettes may only be accessed with a version 2 SpartaDOS loaded into the computer!**

Save Binary File (SAVE and APPEND)

Syntax

```
XIO 41,#IOCB,R,X,"Dn:[path]fname.ext addr1 addr2"
```

Notes
This command will save a binary file between 'addr1' and 'addr2' where 'addr1' and 'addr2' are given in HEX. If R is 8 then the file will be overwritten. If R is 9 then the file will be appended to (as in DOS's APPEND command). If X is less than 128 then a binary file header of $FF $FF will be written; otherwise, it will not be written (preferable for APPENDing segments). Note that the IOCB must not be open. **Atari Diskettes may only be accessed with a version 2 SpartaDOS loaded into the computer!**

Create Directory (CREDIR)

Syntax
XIO 42,#IOCB,0,0,"Dn:path"

Notes
This command creates a new directory. The last name in the pathname is the directory to be created. The path leading up to the name must be a valid and existing path. Note that the IOCB must not be open. **This will not work on Atari DOS 2 diskettes!**

Delete Directory (DELDIR)

Syntax
XIO 43,#IOCB,0,0,"Dn:path"

Notes
This command deletes a directory. The directory must contain no files in order for it to be deleted. The last name in the pathname is the directory to be deleted. The path leading up to the name must be a valid and existing path. Note that the IOCB must not be open. **This will not work on Atari DOS 2 diskettes!**

Change Working Directory (CWD)

Syntax
XIO 44,#IOCB,0,0,"Dn:path"

Notes
This changes the current default directory. The path name must be valid and all directory names in the path must exist. Note that the IOCB must not be open. **This will not work on Atari DOS 2 diskettes!**

Set Boot File (BOOT)

Syntax

XIO 45,#IOCB,0,0,"Dn:[path>]fname[.ext]"

Notes

The IOCB must be closed for this operation to be used. Wild cards may be used in the filename. **This function is valid for only SpartaDOS version 2.x diskettes and must be used with a version 2.x SpartaDOS loaded into the computer!**

Unlock Diskette (UNLOCK)

Syntax

XIO 46,#IOCB,0,0,"Dn:"

Notes

The IOCB must be closed for this operation to be used. **This function is valid for only SpartaDOS version 2.x diskettes and must be used with a version 2.x SpartaDOS loaded into the computer!**

Format Diskette in Atari DOS 2 Format (AINIT)

Syntax

XIO 254,#IOCB,0,0,"Dn:"

Notes

The IOCB must be closed for this operation to be used. **This function may be used only with a version 2.x SpartaDOS load into the computer!**

Directory Listing (DIR)

Syntax

```
10 DIM A$(40):TRAP 40
20 OPEN #IOCB,6,X,"Dn:[path>]fname[.ext]"
30 INPUT #IOCB,A$:PRINT A$:GOTO 30
40 CLOSE #IOCB
```

Notes
If 'X' is less than 128, then a standard Atari DOS 2 listing is given. If 'X' is greater than 127, then the expanded (SpartaDOS) listing is given, showing file size, date and time. For an explanation of the directory format, refer to Chapter 3. **Atari Diskettes may only be accessed with a version 2 SpartaDOS loaded into the computer!**

SpartaDOS User Accessible Data Table

Because SpartaDOS is mainly a command processor driven DOS, a great number of variables have been made user accessible. These are mainly for parameter passing on the command line, time/date interface, addresses of important routines within the DOS, and a few other miscellaneous datum. The data table is referred to as '**COMTAB**' and is pointed to by '**DOSVEC**' (at location 10). A few assembly language routines will follow as an aid. The user area at 'COMTAB' is as follows.

LSIO **[COMTAB-10]**
This location contains the address of the SIO routine SpartaDOS uses. This is actually a vector, so you may replace this address with your own. The Ramdisk patches in here to trap access to the drive it is emulating. Many commands use this vector to run the DOS's high speed SIO routine.

ECHOFLG **[COMTAB-8]**
This location contains the index into HATABS (table of handler ID's and addresses) of the file SpartaDOS is echoing output to. A value of $FF indicates that echoing is inactive. **This location is valid only while running under SpartaDOS 2.x.**

BATFLG **[COMTAB-6]**
This location contains the index into HATABS (table of handler ID's and addresses) of the file SpartaDOS is receiving input from. A value of $FF indicates that no batch file is active. **This location is valid only while running under SpartaDOS 2.x.**

WRTCMD **[COMTAB-2]**
This location contains the SIO write command. A 'W' is the write with verify command, and 'P' is the write with no verify command. **This location is valid only while running under SpartaDOS 2.x.**

WARMST **[COMTAB-1]**

This flag, if set, indicates that the command processor is doing a cold start. It is cleared (to 0) whenever the command processor is entered. It is used to trap errors when trying to open 'STARTUP.BAT' or 'AUTORUN.SYS'.

COMTAB **[COMTAB]**

This location contains a 6502 jump instruction to the command processor. BASIC enters here on a "DOS" command.

ZCRNAME **[COMTAB+3]**

This location contains a 6502 jump instruction to the filename crunch routine ('CRNAME'). This is used by most external DOS commands to fetch the next filename on the command line. The command line is at 'LBUF' and the crunched filename ends up at 'COMFNAM'. This routine supplies the default drive number if necessary. The zero flag on return is SET if no filename is on the command line. Each call returns the next filename on the command line.

ZDIVIO **[COMTAB+6]**

This location contains the address of the divert input/output (redirection of I/O) routine. From an assembly language program, you may call the routine through an indirect jump to 'ZDIVIO' with the filename at 'COMFNAM' and the Y register equal to 0 if output (PRINT), or 1 if input (-fname).

ZXDIVIO **[COMTAB+8]**

This location contains the address of the stop divert input/output routine. From an assembly language program, you may call the routine through an indirect jump to 'ZXDIVIO' with the Y register equal to 0 if stopping output (PRINT), or 1 if stopping input (force end of file).

BUFOFF **[COMTAB+10]**

This location contains the current offset into the command line. 'CRNAME' uses this pointer to fetch the next parameter on the command line (at 'LBUF') and move it to 'COMFNAM'.

ZORIG **[COMTAB+11]**

This location contains the start address of SpartaDOS. $600 is the start address of SPEED.DOS and STANDARD.DOS, and $700 is the start address of all other versions.

DATER **[COMTAB+13]**
This location contains the current date in DD/MM/YY format (3 bytes). This is the date that SpartaDOS inserts in the directory whenever a new file or directory is created. To override this, see 'TDOVER'.

TIMER **[COMTAB+16]**
This location contains the current time in HH/MM/SS format (3 bytes, Not in BCD format, therefore it can be read with no conversion from BASIC). This is the time that SpartaDOS inserts in the directory whenever a new file or directory is created. To override this, see 'TDOVER'.

ODATER **[COMTAB+19]**
This location contains the alternate date in DD/MM/YY format (3 bytes). SpartaDOS uses this date instead of 'DATER' if the 'TDOVER' flag is set.

OTIMER **[COMTAB+22]**
This location contains the alternate time in HH/MM/SS format (3 bytes). SpartaDOS uses this time instead of 'TIMER' if the 'TDOVER' flag is set.

TDOVER **[COMTAB+25]**
This location contains the time/date override flag. It is set to 0 if to use 'DATER' and 'TIMER' when it creates new files, and set to $FF if to use 'ODATER' and 'OTIMER'. This is used by file copy programs (such as SPCOPY and MENU) to insure that the time and date of each file is preserved.

TRUN **[COMTAB+26]**
This location contains the RUN address of a load file. This location is updated during the internal load operation, so BASIC or any other program may check what the load address was. 'RUNLOC' is updated from this location by the command processor only.

[COMTAB+28]
This location always has a value of 128.

DDENT **[COMTAB+29]**
This location contains the density (sector size) of each drive (4 in all — SpartaDOS 1.x only supports 4 drives). A value of 0 indicates 256 byte sectors, and a 128 indicates 128 byte sectors. **This table is valid only for version 1.x SpartaDOS.** The DDENT table is inaccessible if you are using SpartaDOS 2.x.

COMFNAM **[COMTAB+33]**

This is the buffer for the output of the ZCRNAME routine. It is a 28 byte long buffer and **always** begins in the form 'dn:' so if you are only looking for parameters, you may start looking at COMFNAM+3.

RUNLOC **[COMTAB+61]**

This location contains the run address of a '.COM' file when it is loaded through the command processor (as a command). If no address is specified after the RUN command, this is the address to be run.

LBUF **[COMTAB+63]**

This location contains the input buffer. This is where the command line is stored. 'LBUF' is 64 bytes in length.

Format of SpartaDOS Diskettes

Sectors are of four types in SpartaDOS; they are: 1) boot sectors, 2) bit maps, 3) sector maps, or 4) data sectors. Data sectors may be divided into two classes, directory sectors and user data sectors. The following is a detailed description of each sector type and the structure of a SpartaDOS directory.

Sector Maps

Sector maps are simply a list of sectors making up a file. The first two entries are a link to the next sector map and a link to the last (previous) sector map. The rest of the sector is just a list of up to 62 (if SD) or 126 (if DD) data sector numbers.

next:	This is the sector number of the next sector map. It will be a zero if this is the last sector map.
last:	This is the sector number of the last (previous) sector map. It will be zero if this is the first sector map.
data:	These are the sector numbers of the data sectors of the file. If a data sector number is zero, then that portion of the file is not allocated. This can happen if a file is written to at a low file position and then written to at a high file position without ever writing the middle data (see POINT).

Bit Maps

A bit map is a sequence of bits that determine whether a sector is in use or not. Bit 7 represents the first sector in a group of 8 and Bit 0 represents the eighth sector in the group. The first byte of the bit map corresponds to sector numbers 0 through 7, the second corresponds to sector numbers 8 through 15, etc. (NOTE that sector number 0 does not exist). If more than 1 bit map is required for the diskette, they will be sequential on the diskette. A sector is 'free' if the corresponding bit map bit is SET (1).

Boot Sectors

The boot sectors are the first three sectors on all SpartaDOS diskettes. They contain the program that loads in the DOS, links it into the system, and enters the command processor. The first sector contains a large table of data which: points to the first bit map sector, points to the main directory sector map, holds the density and free sectors, and contains the volume name, to name a few. Listed below is where each item of data is kept. The numbers represent offsets into the sector.

9	This is the first sector map of the MAIN directory.
11	This is the total number of sectors on the diskette.
13	This is the number of free sectors on the diskette.
15	This is the number of bit map sectors used on the diskette.
16	This is the sector number of the first bit map sector.
18	This is the sector number to begin the file data sector allocation search. This is the beginning sector number that is checked to see if free when writing a standard file.
20	This is the sector number to begin the directory data sector allocation search. This is the beginning sector number that is checked to see if free when expanding a directory or creating a new directory. This is used so that directory sectors are close together (hopefully continuous) for faster operation.
22	This is the diskette volume name (8 characters). These must be unique on SpartaDOS version 1.x diskettes **or** when using a version 1.x SpartaDOS.
30	This is the number of tracks the diskette has. The most significant bit is set if this is a double sided drive.
31	This is the size of the sectors on this diskette. A 0 indicates 256 byte sectors and a 128 indicates 128 byte sectors.
32	This is the **major** revision number of the DOS on this diskette. Possible values are \$11 (for 1.x versions) and \$20 (for 2.x versions).
33	This is the number of buffers reserved for sector storage (default if booted). **Not applicable to SpartaDOS version 2.x diskettes.**

34 This is the default drive the command processor uses if this diskette is booted.
35 RESERVED
36 RESERVED
37 This is the number of sectors in the main DOS boot (under **version 1.x diskettes only**).
38 This is the volume sequence number. It is incremented every time an open file for write occurs. It is used, in addition to the volume name, to determine if the diskette has been changed. **Only applicable to SpartaDOS version 2.x diskettes.**
39 This is the volume random number. This is simply a random number generated when the diskette was formatted. Its function is the same as the volume sequence number. **Only applicable to SpartaDOS version 2.x diskettes.**
40 This is the first sector map of the file specified by the BOOT command. This is how the boot program knows what file to load. **Only applicable to SpartaDOS version 2.x diskettes.**
42 This is the write lock flag. A value of $FF indicates the diskette is locked, and a 0 indicates that it is not. **Only applicable to SpartaDOS version 2.x diskettes.**

The Directory Data Structure

The directory is a special file that gives information about each file and each subdirectory it contains. Each entry in the directory is 23 bytes in length and contains the filename, the time/date, the length, the first sector map number, and the status of the entry. The first entry is special; it contains the entry describing its directory as a file. The parent directory's entry for a subdirectory maintains everything except the length of the subdirectory. The first entry contains the following information (the numbers are offsets into the entry):

1 This is the first sector map number of the parent directory. A zero indicates that this is the base (or main) directory.
3 This is the length of the directory (3 bytes).
6 This is the directory name (8 bytes).

When a directory is opened (unformatted mode), the file position is automatically set to the second entry. You must do a position if you want to read the entry containing the above information (first entry). The rest of the entries contain the following information (the numbers are offsets into the entry):

0	This is the file status byte. A zero indicates the end of the directory file. The following describes the meaning of **set** bits: B0 - The entry is protected. B3- The entry is in use. B4 - The entry has been deleted. B5 - The entry is a subdirectory.
1	This is the first sector map of the file.
3	This is the length of the file (3 bytes).
6	This is the filename (8 bytes.. space padded).
14	This is the filename extension (3 bytes.. space padded).
17	This is the date the file was created (DD/MM/YY - 3 bytes).
20	This is the time the file was created (HH/MM/SS - 3 bytes).

More SpartaDOS Functions Accessible Through the CIO

The following is a list of special CIO functions **not** included in the list of BASIC XIO functions. These tend to be more difficult to use through BASIC.

Check Diskette Status (CHKDSK)

This functions retrieves information about the diskette. The following are the input and output parameters:

CIO Input Conditions

iccom	=	47
icbal	=	low byte of 'Dn:' address
icbah	=	high byte of 'Dn:' address
icbll	=	low byte of buffer address
icblh	=	high byte of buffer address

CIO Output Results

buffer	=	result of CHKDSK operation (17 bytes)
+0	=	version number of diskette: 0 if Atari DOS 2
+1	=	number of bytes per sector: 0 implies 256
+2	=	total sectors on diskette (low,high)
+4	=	free sectors on diskette (low,high)
+6	=	volume name (8 bytes, SpartaDOS only)
+14	=	volume sequence number (1 byte, SpartaDOS 2.x)
+15	=	volume random number (1 byte, SpartaDOS 2.x)
+16	=	write lock flag (1 byte, 0=false (unlocked), SpartaDOS 2.x)

Get Current Directory Path (?DIR)

This function returns the current directory path. The input and output parameters are as follows:

CIO Input Conditions

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

CIO Output Results

buffer	=	result of ?DIR operation. This is a legal path describing path to the directory specified in the command. If no path is specified, then the function returns the current default directory path. The path is ended with an EOL.

CHAPTER 20 — DIFFERENCES BETWEEN SPARTADOS 1.x AND 2.x

This chapter lists the enhancements written into the CP version 2.x of SpartaDOS. It is intended mainly for those who have already been using version 1.x that want to quickly know the differences. The new external commands are not listed here. Use the table of contents or the command summary to review those. The major differences between the two versions are as follows:

1. SpartaDOS 2.x resides primarily in the RAM underneath the OS ROM of the computer. Therefore, it will only work on the XL/XE computers. (Since a standard 800 doesn't have RAM under the OS). By using this method, you now have about 4K more usable memory available.

2. SpartaDOS can now read and write all Atari DOS 2 diskettes automatically. If an Atari DOS 2 diskette is in the disk drive, all functions that work with Atari DOS 2 will work **exactly** as they did while using Atari DOS 2. There are a few additions to the Atari handler as follows:

   a. A CHKDSK XIO function has been added to the Atari handler (described in Chapter 19).

   b. The Write capability has been enhanced. In UPDATE mode, you may continue writing beyond the end of the file. Before you could only write up to the end; an error would result if you tried to write further. Now you automatically enter an append mode when the transition is made. In essence, file operations work the same way as in the SpartaDOS handler with the exception of NOTE/POINT. These preserve the Atari DOS 2 interpretation (sector #/offset).

   c. The disk initialization function (XIO 254) has also been included. This will format exactly like Atari DOS 2 would. The density is dependent upon the configuration of the drive as is normal with all Atari DOS 2 implementations.

   d. SpartaDOS will also work with double density Atari DOS 2 diskettes.

   e. You may now open files after the formatted directory has been opened. Atari DOS 2 has a bug where it loses its place in the directory when a file is opened.

3. All references to COPY being in page 6 should be ignored. COPY is now completely internal and resides under the OS ROM.

4. The BUFS command has been eliminated. There are now always 12 buffers which reside under the OS ROM.

5. Most errors that can occur with use of the command processor have error messages displayed rather than error numbers.

6. Unique volume names are no longer required (as long as you are using version 2.x). A random number is put on the boot sector (sector 1) during format time. A sequence number is also used, and is incremented every time a file is closed that was open for writing, or whenever any disk modifying command is performed (i.e. ERASE, CREDIR, etc.).

7. Batch files may now be linked (i.e. the last line in a batch file may call another batch file). Also, PRINT is no longer a toggle. Without any parameters, PRINT will just close the current file. With a filename, it will close the current file (if one was open), and then start a new PRINT file. NOTE: these changes are at the XDIVIO/DIVIO level, so if using the assembly calls, this will work.

8. The PAUSE and MEM commands are now internal.

9. A file with one or more wild cards cannot be written to the diskette. Thus, the command 'COPY E:' will not be allowed. This protects against accidental erasures of the first file. (Before, this command would replace the first file and take on a name of '????????.???'. The only way to erase this file was to 'ERASE *.*'.)

10. PROTECT/UNPROTECT commands have been added. They work identically to the Atari DOS 2 commands. The command syntax is:

```
PROTECT [Dn:]fname[.ext]
UNPROTECT [Dn:]fname[.ext]
```

The XIO codes are the same as described by Atari DOS 2. (protect=35, unprotect=36)

9. Diskettes may be software LOCKed and UNLOCKed. If a diskette is locked, it will act just as though a write protect tab is on the diskette, however a different error code is returned. The command syntax is:

   ```
   LOCK [Dn:]
   UNLOCK [Dn:]
   ```

   The XIO codes for these commands are: lock=34, unlock=46.

10. The following error codes and meanings have been added:

$95	=	Not version 2.x diskette
$94	=	Not a SpartaDOS diskette
$A3	=	Illegal wild card in filename
$A4	=	File protected (attempt to replace was made)
$A9	=	Diskette is write locked

11. The VERIFY command has been added. This allows the user to change between write with verify and write without verify. The syntax of the command is:

    ```
    VERIFY ON or
    VERIFY OFF
    ```

12. The XDIV command has been added. This command permanently disables the batch file and PRINT capability. This is because some application programs will function incorrectly while the EDITOR handler is patched. The command syntax is:

    ```
    XDIV
    ```

13. The DIRS directory command has been added. This will list the directory in Atari DOS 2 compatible mode. The sectors per file field is a calculated number and may be 1 off but is unlikely with files under 8K. This type of directory listing was previously available, but not under the command processor, nor did it have correct sector counts; they were zero filled before.

14. An error will be given if SpartaDOS 2.x is read into a non-XL Atari computer. RESET will reboot the computer. Also, an error will be given if there is no DOS (as set by BOOT or XINIT) on the diskette.

15. The BOOT command has been added. This command will select a program (normally DOS) to load when the diskette is booted. The file selected must be a standard binary load file. The INITZ and RUN vectors are handled normally (as described under Atari DOS 2 and SpartaDOS). The syntax of the command is:

    ```
    BOOT [Dn:]fname[.ext]
    ```

    This is the XIO function number 45.

16. The DOS loader (on the first 3 sectors of each diskette) can now load files in the same manner as the LOAD command. Normally DOS is loaded, but actually anything could be loaded. This makes a good way of creating binary boot programs. NOTE: the loader resides from $3000-$3180 and uses $2E00-$3000 for data, so the booted program must not overwrite these areas.

17. The MAIN directory is no longer scanned twice when OPENed for READ, if the CURRENT directory is the MAIN directory. Before, when trying to load a nonexistent file at the base (MAIN) directory, the directory was scanned twice for the file.

18. From assembly language, the character immediately following the filename does not have to be a $9B or less than $20. Now any non-alphanumeric character can end a filename except '>', which is reserved as a place holder in pathnames. This is so a few more programs will work correctly. Also, a comma may delimit filenames for the RENAME function. (MEDIT uses a comma)

19. The CAR command now checks to make sure a cartridge is present.

20. A CHKDSK command has been added. It states the volume name, the random and sequence numbers, the total bytes/disk, the free bytes/disk, the sector size, and the write lock status. Only SpartaDOS 2.x diskettes will display the write lock status and the sequence and random numbers. This is also an XIO function (described in Chapter 19). The command syntax is:

    ```
    CHKDSK [Dn:]
    ```

21. The following data structures and definitions have been modified:

    a. BIT[0] of the directory status byte is SET (1) if the entry is PROTECTED, and CLEAR if not.
    b. at +38 in sector 1 is the volume sequence number.
    c. at +39 in sector 1 is the volume random number.
    d. at +40 in sector 1 is the first sector map number of the file to boot (i.e. XD23B.DOS etc).
    e. at +42 in sector 1 is the write lock flag; $FF is locked.

22. The AINIT command has been added. This is the Atari DOS 2 format command, and is also an XIO function. A Yes/No prompt will be given to make sure. The syntax is:

    ```
    AINIT [Dn:]
    ```

    The XIO function number of format is 254.

23. The following are new definitions of offsets within COMTAB (numbers represent offsets):

    -2SIO command used to write a sector ('P' or 'W')
    -7Flag indicating active DIVIN ($FF is false)
    -8Flag indicating active DIVOUT ($FF is false)

24. An error message ('File not found') will be given if a RENAME, ERASE, PROTECT, or UNPROTECT command is used and no files match the filespec.

25. High Speed I/O routines are placed under the OS ROM. During boot up of the diskette, the high speed routines are automatically switched to as soon as they are loaded.

APPENDIX A — ERRORS

Atari Basic Error Messages

CODE #	ERROR CODE MESSAGE
2	Insufficient Memory
3	Value Error
4	Too Many Variables
5	String Length Error
6	Out of Data Error
7	Number > 32767
8	Input Statement Error
9	Array or String DIM Error
11	Floating Point Overflow/Underflow Error
12	Line Not Found
13	No Matching FOR Statement
15	GOSUB or FOR Line Deleted
16	RETURN Error
17	Garbage Error
18	Invalid String Character
19	LOAD Program Too Long
20	Device Number >7 or =0
21	LOAD File Error

SpartaDOS Error Messages

CODE #	ERROR CODE MESSAGE
128($80)	BREAK Abort
129($81)	IOCB Already Open (Input/Output Control Block)
130($82)	Nonexistent Device Specified - You typed an undefined device. Legal devices are: D:,S:,C:,R:,P:,E:.
131($83)	File/IOCB Not Open For Read
132($84)	Invalid IOCB Command
133($85)	Device or File/IOCB Not Open
134($86)	Bad IOCB Number
135($87)	File/IOCB Not Open For Write
136($88)	End of File
137($89)	Truncated Record
138($8A)	Device Timeout (No Drive Found)
139($8B)	Device NAK (Not Acknowledged) This is a message you'll get when trying to read an incompatible DOS or disk not in place.
144($90)	Device Done Error (Bad Sector/Disk Write Protected)
146($92)	Function Not Implemented in Handler

148($94)	Not a SpartaDOS Diskette
149($95)	Diskette not SpartaDOS version 2.x
150($96)	Directory Not Found
151($97)	File Exists. May not replace or delete file. Can happen when saving a file with a directory of the same name (dname = fname.ext).
152($98)	Not a Binary File
160($A0)	Drive Number Error
162($A2)	Disk Full (no free sectors)
163($A3)	Illegal Wild Card in Filename
164($A4)	File Erase Protected
165($A5)	File Name Error - Typed illegal characters in filename.
166($A6)	Position Range Error
167($A7)	Cannot Delete Directory
168($A8)	Illegal DOS Command / Not Implemented
169($A9)	Diskette is Write Locked
170($AA)	File Not Found - You've mistyped a file or command name or tried a write operation with NOWRITE.DOS.

APPENDIX B - COMMAND SUMMARY

?DIR [Dn:][path] Internal - 2.x
To show the path to a specified directory. If no path is given as a parameter, the current directory path is displayed.

AINIT [Dn:] Internal - 2.x
This command is used to format a diskette in Atari DOS 2 style format.

APPEND [Dn:][path>]fname[.ext] address address Internal - 1.x and 2.x
This command saves a binary block of data at the end of an existing binary file.

BASIC ON or BASIC OFF Internal - 2.x
This command installs or removes the internal BASIC with the XL/XE computers.

Batch Files (syntax below)
-fname 1.x and 2.x
To retrieve and execute a file (**fname.BAT**) which instructs DOS to go perform specific operations in a specific order. **STARTUP.BAT** is a special batch file which is automatically executed when the diskette is booted.

BOOT [Dn:][path>]fname[.ext] Internal - 2.x
This command tells a SpartaDOS 2.x formatted disk to boot a particular program at startup.

BUFS [n] Internal - 1.x
To set or check the number of buffers currently in use under CP version 1.x only.

CAR Internal - 1.x and 2.x
Exit from DOS to a language cartridge.

CHKDSK [Dn:] Internal - 2.x
To display the volume name, random & sequence numbers (version 2.x diskettes only), sector size, formatted bytes on disk, available bytes on disk, and write lock status (version 2.x diskettes only).

CHTD [Dn:][path>]fname[.ext] External - 1.x and 2.x
This utility command is used to change a file's time/da

CHVOL [Dn:]vname External - 1.x and 2.x
This utility command is used to change the volume name on a diskette.

Command Files (syntax below)
[Dn:][path>]fname [parameters]1.x and 2.x
To load and run binary files. Also, it also provides a standard for passing parameters to machine language programs.

COPY d[n]:[path>][fname[.ext]] [dn:][path>][fname[.ext]][/A]
Internal - 1.x with exceptions and 2.x
Note: the '/A' option is allowed under CP version 2.x only.

COPY one or more files from one device to another and if specified, gives the copy a different name. COPY can also be used to append one file to another under CP version 2.x only.

COPY can copy files to the same disk, however the copy must have a different name unless the destination is another directory. Note that a file may NOT be copied to the same disk drive with a different diskette. There is no provision to switch diskettes in the middle of the COPY process. If a single drive copy is desired, see the SPCOPY, XCOPY, MENU or DUPDSK commands.

You may also use COPY to transfer data between any of the other system devices, i.e. the Screen Editor, Printer, Keyboard, etc.

CREDIR [Dn:]path Internal - 1.x and 2.x
Creates a subdirectory on the specified disk.

CWD [Dn:]path Internal - 1.x and 2.x
Change the working (current) directory on the specified disk.

DELDIR [Dn:]path Internal - 1.x and 2.x
Deletes an empty subdirectory from the specified disk.

DIR [Dn:][path>][fname[.ext]] or
DIRS [Dn:][path>][fname[.ext]] (optional with CP version 2.x)
Internal - 1.x and 2.x with exceptions

To display the volume name and the specified directory name, to list files and subdirectories in the directory, the file size in bytes, the date and time the files were created, and the number of free sectors left on disk. DIR may be used to list all files matching a file spec pattern by using wild cards. DIRS displays the short form directory as used with Atari DOS 2.0 (CP version 2.x only)

DIS__BAT External - 1.x and 2.x (use XDIV with 2.x)
The DIS__BAT command is used with CP version 1.x to disable batch processing and the PRINT command (redirection of I/O). This may be necessary in order to run certain programs. If using CP version 2.x see the XDIV command.

DUMP [Dn:][path>]fname[.ext] [start [#bytes]] [/P]External - 1.x and 2.x
This utility will display a file or portion of a file in HEX and ATASCII or ASCII format.

DUPDSK External - 1.x and 2.x
To duplicate an entire SpartaDOS diskette (except for volume name) using one or two disk drives. Note: Number of tracks and densities must match on 'source' and 'destination' disks or an error will result.

ERASE [Dn:][path>][fname[.ext]] Internal - 1.x and 2.x
ERASE deletes the file or files from the specified file name from the specified directory. If no path is specified, the file is deleted from the current directory.

FORMAT External - 1.x and 2.x
This command is used to format the diskette, create the directory structure, and optionally put DOS on the diskette. (Only 1.x diskettes.)

INIT External - 1.x and 2.x
This is the master formatting program for SpartaDOS 1.x versions and allows selection of certain default parameters. (Will create 1.x versions.)

KEY External - 1.x (for 400/800 computers)
XKEY External - 1.x and 2.x (for XL/XE computers)
To install a 32 character keyboard buffer.

LOAD [Dn:][path>]fname[.ext] Internal - 1.x and 2.x
This command loads any binary file into memory but does not run the file. The standard DOS RUN and INIT vectors are **not** used.

LOCK [Dn:] and **UNLOCK [Dn:]** Internal - 2.x

The LOCK command locks the disk to prevent accidental erasure. It is similar to the physical write protect tab which is put on the disk, but is strictly a software lock and only works when using CP version 2 diskettes. UNLOCK disables the LOCK command. (Only affects 2.x diskettes.)

MDUMP [address [#bytes]] [/P] External - 1.x and 2.x

This utility will display memory locations in HEX and ATASCII or ASCII format. It is very similar to DUMP but works on blocks of memory rather than files.

MEM Internal - 2.x

MEMLO External - 1.x and 2.x (2.x should use MEM)

To display MEMLO (lower bound) and MEMHI (upper bound — **only MEM displays this**).

MENU [R][n] External - 2.x

This command gives you most of the features of the command processor but in a menu form. It is capable of single and multiple file functions.

OFF LOAD [Dn:][path>]fname[.ext] offset [/SNPQ] or

OFF LOAD -R address [Dn:][path>]fname[.ext]

External - 1.x and 2.x

This utility command loads in files at an offset and optionally displays segment addresses, file position of beginning of segment, and can query whether to load a given segment. If may also be used to create non-relocatable versions of OFF__LOAD.

PAUSE External - 1.x, Internal 2.x

To temporarily halt execution of a batch file and to prompt the user for a response to continue.

PORT [path>]fname[.ext] External - 1.x and 2.x

To set speed, word size, stop bits, translation, input and output parity, and EOL parameters for RS232 communications.

PRINT [dn:][path>]fname[.ext][/A] or **PRINT d[n]:** or **PRINT**

Internal - 1.x and 2.x

Note: the '/A' option is allowed under CP version 2.x only.

To echo all output that is written to the screen editor (E: through IOCB #0) to a specified output device.

PROTECT [Dn:][path>]fname[.ext] or Internal - 2.x
UNPROTECT [Dn:][path>]fname[.ext]
These commands protect and unprotect (lock & unlock) files from accidental erasure. (This only affects 2.x diskettes.)

PUTRUN [Dn:]fname[.ext] External - 1.x and 2.x
This command appends the RUN vector containing the start address of an external command file to the file. This is to make a command such as MENU be able to run as an AUTORUN.SYS (when only RUN/INIT vectors are used).

RDBASIC Dn: (XL/XE computer with internal BASIC on required)
RD130 Dn: (Atari 130XE computer required)
RDAXLON Dn: (Axlon RAMPOWER 128 in Atari 800 required)
External - 1.x and 2.x with restrictions

These commands install a Ramdisk device (electronic disk) in the place of a disk drive. Since these commands depend on specific hardware, the correct device must be present or an error will result. Note: CP version 1.x allows up to 4 drives and CP version 2.x allows up to 8 drives.

RENAME [Dn:][path>]fname[.ext] fname[.ext] Internal - 1.x and 2.x
Change the name of an existing file or files.

RPM [Dn:] External - 1.x and 2.x
To display the drive speed in RPM for user information.

RS232 External - 1.x and 2.x (for the 850 interface)
AT_RS232 External - 1.x and 2.x (for the ATR8000)
To load the RS232 Handler for communications.

RUN [address] Internal - 1.x and 2.x
To re-execute the last '.COM' file or execute at a given address. (To load and run a binary file see 'Command Files'.)

SAVE [Dn:][path>]fname[.ext][/A] address address Internal - 1.x and 2.x the '/A' option is allowed under CP version 2.x only.

This command saves binary data from memory to disk. To append data, see the APPEND command, or with CP version 2.x use the '/A' option.

SET [mm/dd/yy] [hh/mm/ss] (for use with TIME command)
TSET [mm/dd/yy] [hh/mm/ss] (for use with TD or XTD commands)
External - 1.x and 2.x
These commands allow the user to set the time and date after installing the clock with the TIME, TD or XTD commands.

SPCOPY or **XCOPY** External - 1.x and 2.x
These commands are used for single or dual drive file transfers between SpartaDOS and/or Atari DOS 2 compatible formats with few restrictions on density and number of tracks. This is the way to convert Atari DOS 2 files to SpartaDOS or the reverse of this. Since translation is already built into CP version 2.x, use the smaller XCOPY with that version of DOS.

TD [X] (for use with R-TIME 8 cartridge)
TIME [X] (for use with system clock)
XTD (for use with R-TIME 8)
External - 1.x and 2.x

TD and XTD are used with ICD's R-TIME 8 Cartridge to install the hardware clock. XTD installs the R-TIME 8 without a display and TD installs it with the date and time displayed at the top of the screen. TIME installs the clock built into the Atari which is not very accurate and must be set upon system boot. The 'X' parameter will turn the time and date display off but keep the clock installed.

TREE [Dn:][path] [/F] External - 1.x and 2.x
To display all the directory paths found on the disk or under the specified directory, and to optionally list the files found in each directory in alphabetical order.

TYPE [Dn:][path>]fname[.ext] Internal - 1.x and 2.x
To display the contents of an ASCII file. Commonly used to read a batch file without executing it. Does not disturb the contents of memory like COPY to E:.

UNERASE [Dn:][path>][fname[.ext]] External - 1.x and 2.x
To restore a file that has been erased.

VERIFY ON or **VERIFY OFF** Internal - 2.x
This command changes the write mode to write with verify or write without verify.

XDIV Internal - 2.x

The XDIV command is used with CP version 2.x to disable batch processing and the PRINT command (redirection of I/O). This may be necessary in order to run certain programs. **If using CP version 1.x see DIS__BAT**.

XINIT External - 1.x and 2.x

This is the command to initialize (format) a SpartaDOS 2.x diskette.

APPENDIX C—Table of all SpartaDOS Command Processor Commands

Command Name	Internal Command in V1.x	Internal Command in V2.x	Works with a DOS 2 disk	External SpartaDOS Command	May use RUN to re-enter	Command locates itself at MEMLO	Function remains resident at MEMLO	Initial Load Address
?DIR	NO	YES	NO	NO	—	—	—	—
AINIT	NO	YES	—	NO	—	—	—	—
APPEND	YES	YES	YES	NO	—	—	—	—
AT__RS232	NO	NO	—	YES	NO	YES	YES	$5000
BASIC	NO	YES	—	NO	—	—	—	—
BOOT	NO	YES	NO	NO	—	—	—	—
BUFS	YES	NO	—	NO	—	—	—	—
CAR	YES	YES	—	NO	—	—	—	—
CHKDSK	NO	YES	YES	NO	—	—	—	—
CHTD	NO	NO	NO	YES	NO	NO	NO	$5000
CHVOL	NO	NO	NO	YES	NO	NO	NO	$5000
COPY	YES	YES	YES	NO	—	—	—	—
CREDIR	YES	YES	NO	NO	—	—	—	—
CWD	YES	YES	NO	NO	—	—	—	—
DELDIR	YES	YES	NO	NO	—	—	—	—
DIR	YES	YES	YES	NO	—	—	—	—
DIRS	NO	YES	YES	NO	—	—	—	—
DIS__BAT	NO	NO	—	YES	YES	NO	NO	$5000
DUMP	NO	NO	(1)	YES	NO	NO	NO	$5000
DUPDSK	NO	NO	NO	YES	YES	YES	NO	$5000
ERASE	YES	YES	YES	NO	—	—	—	—
FORMAT	NO	NO	—	YES	YES	NO	NO	$6000
INIT	NO	NO	—	YES	YES	NO	NO	$6000
KEY	NO	NO	—	YES	NO	YES	YES	$5000
LOAD	YES	YES	YES	NO	—	—	—	—
LOCK	NO	YES	NO	NO	—	—	—	—
MDUMP	NO	NO	—	YES	NO	NO	NO	$5000
MEM	NO	YES	—	NO	—	—	—	—
MEMLO	NO	NO	—	YES	YES	NO	NO	$5000
MENU	NO	NO	YES	YES	NO	YES	(2)	$5000
OFF__LOAD	NO	NO	(3)	YES	NO	YES	NO	$B400
PAUSE	NO	YES	—	YES	YES	NO	NO	$5000
PORT	NO	NO	—	YES	NO	NO	NO	$5000
PRINT	YES	YES	YES	NO	—	—	—	—
PROTECT	NO	YES	YES	NO	—	—	—	—

Table of all SpartaDOS Command Processor Commands (continued)

Command Name	Internal Command in V1.x	Internal Command in V2.x	Works with a DOS 2 disk	External SpartaDOS Command	May use RUN to re-enter	Command locates itself at MEMLO	Function remains resident at MEMLO	Initial Load Address
PUTRUN	NO	NO	YES	YES	NO	NO	NO	$5000
RD130	NO	NO	—	YES	NO	YES	YES	$3C00
RDAXLON	NO	NO	—	YES	NO	YES	YES	$3C00
RDBASIC	NO	NO	—	YES	NO	YES	YES	$5000
RENAME	YES	YES	YES	NO	—	—	—	—
RPM	NO	NO	—	YES	NO	NO	NO	$5000
RS232	NO	NO	—	YES	NO	YES	YES	$5000
RUN	YES	YES	—	NO	—	—	—	—
SAVE	YES	YES	YES	NO	—	—	—	—
SET	NO	NO	—	YES	YES	NO	NO	$5000
SPCOPY	NO	NO	YES	YES	YES	NO	NO	$3000
TD	NO	NO	—	YES	NO	YES	YES	$5000
TIME	NO	NO	—	YES	NO	YES	YES	$5000
TREE	NO	NO	NO	YES	NO	NO	NO	$5000
TSET	NO	NO	—	YES	YES	NO	NO	$5000
TYPE	YES	YES	YES	NO	—	—	—	—
UNERASE	NO	NO	NO	YES	NO	NO	NO	$5000
UNLOCK	NO	YES	NO	NO	—	—	—	—
UNPROTECT	NO	YES	YES	NO	—	—	—	—
VERIFY	NO	YES	—	NO	—	—	—	—
XCOPY	NO	NO	(4)	YES	YES	YES	NO	$5000
XDIV	NO	YES	—	NO	—	—	—	—
XINIT	NO	NO	—	YES	YES	NO	NO	$4180
XKEY	NO	NO	—	YES	NO	YES	YES	$5000
XTD	NO	NO	—	YES	NO	YES	YES	$5000

1) Cannot use a start offset when dumping a file from an Atari DOS 2 type disk.
2) Remains resident when using the [R] parameter.
3) Cannot use N or Q options with Atari DOS 2 type disk.
4) Only with version 2.x SpartaDOS in system.

APPENDIX D — HOW TO ACCESS THE REAL TIME CLOCK

SpartaDOS keeps the internal time/date clock running and stores the values in memory. These can be used in your applications programs whenever access to time or date is desired. The values are stored in COMTAB+13 to COMTAB+18. The pointer to COMTAB is stored at DOSVEC (locations 10 and 11).

COMTAB+13 =	location of day
COMTAB+14 =	location of month
COMTAB+15 =	location of year
COMTAB+16 =	location of hours (24 hour format)
COMTAB+17 =	location of minutes
COMTAB+18 =	location of seconds

The BASIC program below will display these values. It was written as a plain and simple example for those starting out in BASIC or new to programming the Atari. To read the time/date values use PEEK and to change the values use POKE.

```
10 CMTAB=PEEK(10)+PEEK(11)*256
20 FOR T=13 TO 18
30 ? PEEK(CMTAB+T)
40 NEXT T
```

Note: A special SpartaDOS handler is used with the TD, XTD, and TSET commands to access our optional R-TIME 8 Clock/Calendar Cartridge. This automatically updates the internal real time clock used by DOS. Since the cartridge is very difficult to read directly, we recommend you read it with the proper handler installed through the DOS locations as shown in the above example.

APPENDIX E — ATARI DOS 2 VS SPARTADOS

Atari DOS 2 Menu and SpartaDOS Equivalents

ATARI DOS 2.0S		SpartaDOS Semi Equivalent
A.	**DISK DIRECTORY**	**DIR**
B.	**RUN CARTRIDGE**	**CAR**
C.	**COPY FILE**	**COPY**
D.	**DELETE FILE**	**ERASE**
E.	**RENAME FILE**	**RENAME**
F.	**LOCK FILE**	**PROTECT**(SEE UNERASE)
G.	**UNLOCK FILE**	**UNPROTECT**
H.	**WRITE DOS FILES**	(SEE FORMAT/INIT/XINIT)
I.	**FORMAT DISK**	**FORMAT/XINIT/INIT/AINIT**
J.	**DUPLICATE DISK**	**DUPDSK**
K.	**BINARY SAVE**	**SAVE/APPEND**
L.	**BINARY LOAD**	**LOAD/OFF_LOAD/fname**
M.	**RUN AT ADDRESS**	**RUN**
N.	**CREATE MEM.SAV**	(NOT NEEDED WITH RESIDENT DOS)
O.	**DUPLICATE FILE**	**SPCOPY/XCOPY**

SpartaDOS Commands With no Atari DOS 2 Equivalent

?DIR	AT_RS232	BASIC	Batch Files	BOOT
BUFS	CHKDSK	CHTD	CHVOL	CREDIR
CWD	DELDIR	DIS_BAT	DUMP	FORMAT
INIT	KEY	LOCK	MDUMP	MEM
MEMLO	OFF_LOAD	PAUSE	PORT	PRINT
PUTRUN	RD130	RDAXLON	RDBASIC	RPM
SET	SPCOPY	TD	TIME	TREE
TSET	TYPE	UNERASE	UNLOCK	VERIFY
XCOPY	XDIV	XINIT	XKEY	XTD

A Few Other Major Advantages

SpartaDOS supports all densities and possible configurations for the Atari Computer line. There is no need to configure your drive for a particular density as it automatically checks format when reading.

Time and date stamping is available for all files including support of our hardware real time clock (R-Time 8).

UltraSpeed I/O is supported with appropriate drive hardware.

RS-232 Handlers are included for communications using the Atari 850 interface or the ATR8000 Computer interface.

The SPCOPY command allows file transfer in batches from any density to any density using one or two drives and automatically translates in both directions to or from SpartaDOS. SpartaDOS version 2.x even includes the full Atari DOS 2 handler with several added features.

SpartaDOS has full subdirectory support.

. . .and much much more!

APPENDIX F — US DOUBLER INSTALLATION

Brief Overview

The US Doubler consists of two plug in modules which are to be installed in your Atari 1050 disk drive. One of these is a 24 pin chip (U10) and the other is a hybrid 24 pin module (U8). These are to be installed into the corresponding sockets on the 1050's Printed Circuit Board (PCB). Atari is currently selling 1050s with two different types of U10 chips. The replacement U10 supplied by ICD, is the most common type found. If it is the wrong type for your drive, you can either move two jumpers (which requires soldering), or send us your ICD - U10 for an exchange with the other type.

Before Installing

Be sure to fill out your warranty card and mail it in. This is the only way we will be able to notify you of changes and updates, and the only way you will be eligible for upgrades. Please, take the time to fill in your Atari dealer's name and address, so we can make him aware of our products for the Atari.

If after reading these instructions you feel this installation is not for you, then talk to your local dealer or service center about it, or send us the drive. ICD will install this product for $15.00 including UPS ground shipping one way. This low price is good only before you attempt to install the US Doubler. For later services see our prices at the end of this appendix. For installation by ICD, send and mark the box to:

ICD, Inc.
1220 Rock Street
Rockford, IL 61101-1437

Attn: 1050 Install

Please include a check for $15.00, the complete drive less cable and power supply, and the ICD product. Our turnaround is generally 48 hours.

Do You Still want to install it?

Tools needed

#2 Phillips head screwdriver
#1 Phillips head screwdriver (for some drives made in Hong Kong)
Medium or small flat blade screwdriver
A permanent ink marking pen for marking connectors during disassembly
An empty dish for holding parts
A clean well lighted work surface
Small needlenose pliers
20-35 watt small tipped soldering iron (optional)

Let's Get Started!

Cover removal

Turn the 1050 on its back and remove the 6 phillips head screws. (4 are recessed and 2 are on the front bezel.) Place the screws into your parts dish.

Carefully turn the drive back onto its feet and set it down. Lift the rear of the top cover about 1/2 inch then slide it towards the front and lift the cover and bezel off as one piece. Set these aside.

Things to look for

Notice how the drive assembly sits in the case and note the four black rubber washers under the drive frame. These usually fall out when removing the drive. (Some Hong Kong drives have these glued down.) There are also four steel pins at the center of these washers which fall out during disassembly of the early 1050 drives (they are glued in on the later drives). Notice the wires which connect the drive to its PCB towards the rear. These should all be marked with J14, J10, etc. on the connectors. The markings correspond with markings on the PCB but they don't always indicate the proper polarity. Take your marker and draw a line across the inside of each connector. We will then know when we plug them back in that the side with the black line goes towards center. Do the same on all other connectors (there is one under the front of the drive frame). We are now ready for the heavy work.

Important: Some Hong Kong drives have connectors with no markings and color coded wires. If this is your situation you willl need to make a chart indicating the color pattern for each connector before you unplug them.

Here Goes Nothing . . .

Remove the drive (optional)

Actually, it is not always necessary to unplug the wires from the PCB. You can leave the drive plugged to the board as long as you are very careful with the wires. They are small and will break if too much stress is applied. If you choose to leave the drive plugged in (we usually do), then proceed to **remove the PCB**, otherwise, read on.

Carefully unplug all seven connectors while noting their positioning. **Don't** pull on the wires ; **Do** pull on the plastic connectors. A small needlenose pliers can make this easier for tight fitting connectors. After removing the wires, lift the drive frame up and out of the case and set it aside. Put the four rubber spacers and the four steel pins (if they're loose) in your parts dish.

At last the PCB!

Remove the Printed Circuit Board (PCB)

You are now looking at the PCB. The Chips (ICs) to be replaced are under that large tin cover (shield) which is fastened on the foil side (the bottom side) of the PCB with twisted metal tabs. This shield was designed to reduce RFI (interference with TVs, radios, etc.) The PCB is held down to the case with either, four plastic tabs, or two plastic tabs, or three small phillips head screws and three brown insulating washers. If you have screws holding the board down (most Hong Kong drives do), remove these first. If you have tabs, I find it easiest to lift the front of the PCB while bending the tabs with my other hand. The PCB needs to go slightly towards the front then out of the case! Place the PCB with its component side down on your work area. (If the drive is still attached to the PCB you begin your balancing act.)

Remove the metal shield

The bottom shield on the foil side of the PCB is symmetrical but the top shield has a notched out area in one corner. This notch is for clearance of the solder connections on components R43 and U14. Straighten the tabs and remove the two shields. Turn the PCB over, component side up, and get ready for fun. (If the drive is attached you are lifting it off the board with one hand while working with the other.)

Remove the old ICs

The two 24 pin ICs, U8 and U10, must be removed. Use the flat bladed screwdriver and gently pry the chips out of their sockets and set them aside. These two will not be used again.

★ *Check the jumpers* ★

This is the most important installation step and where most mistakes are made, so pay attention! JP1 through JP7 are the jumper wires behind U10 (See diagram). In most installations, only some of the JP (jumper) numbers will be visible. The other numbers are usually hidden under the jumpers themselves. These jumpers might be solid pieces of wire soldered between two pads or a wire with a white ceramic covering around the center or they might look just like resistors. It does not matter which type is installed; they all serve the same purpose. The position of the first four jumpers (JP1-JP4) determines which type of U10 chip you will need. We're not really sure why Atari used the jumper system when the 1050 drive was designed. Maybe it was so they could switch chip types when one became more cost effective. There are many manufacturers of both types of CHIPs and each works as well as the other for this application. The only difference is pin configuration, which is what the jumpers change.

If the replacement U10 **has** a paper label on it, then JP1 and JP3 should be open (no connection) and JP2 and JP4 should be closed (jumpered). If the replacement U10 does **not** have a paper label then JP2 and JP4 should be open (no connection); JP1 and JP3 should be closed (jumpered). Every effort has been made by ICD to provide you with the most common type of U10 chip. Recently (May 1985) we have found that most drives 'Made in Singapore' need the U10 without the paper label, and most 'Made in Hong Kong' need the U10 with the label. The U10 which comes with the drive will usually also either have a paper label or not; this should match the corresponding ICD U10 needed. The only sure way to tell is to check those jumpers!

If your replacement U10 is of the wrong type, you have two options. 1) Send us the ICD U10 (in protective packing) along with $1.00 for shipping and handling and mark on the outside 'Attn: U10'. When we receive this, we will send the other type of U10 which you can then plug in. 2) Move the jumpers to the correct locations for the ICD U10 chip in your posession. Do not attempt this modification unless you feel confident with a soldering iron. The other jumpers JP5 through JP7 should always remain unmoved.

Plug in the chips
For correct positioning, the notches at the ends of the modules (chips) go towards the front of the drive. Also, as a general rule, any labels or writing on your ICD replacement chips will read from the front of the drive to the rear. Now carefully plug the new U8 (the larger module) into the socket for U8. Next, carefully plug the new U10 into its socket with the notch towards the front of the PCB. Make sure all the pins went into the correct holes in the sockets. Wasn't that easy?

REASSEMBLY

Put the shield back on
If you're unsure of what you are doing then you might want to leave the metal shield off for testing. If you haven't had any problems following us so far then it's all down hill from here. Be careful installing the shield and make sure the notched end of the top piece is over R43 and U14. Also make sure that no components or wires are pinched between the shield and the PCB.

Put the PCB back into the case
Place the rear in first, then lower the front of the PCB. The PCB should easily snap in place under the plastic tabs. (Install the three washers and screws if your drive had them.)

Reinstall the rubber washers and steel pins (if removed)
Press the four rubber washers with the recessed side down, onto the plastic posts in the front half of the drive's case. The four steel pins are either still stuck in the plastic posts or if they were loose, (older drives) take them from your parts dish and put one into each hole at the center of the rubber washers.

Reinstall the drive frame
Plug the connector from the drive head onto J6 at the front of the PCB. Carefully lower the drive frame onto the steel pins noting that the steel pins fit into holes in the drive frame.

Plug in the connectors (if unplugged)

Plug the rest of the connectors onto the corresponding pin locations. Be sure to note the marking you made on the connectors during disassembly. (If you did not unplug your drive from the PCB, you can skip this instruction. That's your reward for being so brave and talented.)

Replace the top cover

To replace the top cover, first line up the bezel over the front of the drive frame, then lower the cover. If the bezel becomes separated, put the top cover on first, then hook the top of the bezel under the top cover front edge, and gently snap it down into place. While holding the case together turn the disk drive upside down and lay it on its back. Screw the six phillips screws back into place and presto!

You're Done!

STARTUP AND TESTING

Plug the drive back into your system. If you're going to use UltraSpeed (US), it is better to make this drive number one, so you can boot up from this drive. Put a SpartaDOS Master disk into the drive, close the door, and power up the computer. If you get an error message 'Not an XL/XE computer' then use the other Master disk. Impressed? The MASTER SpartaDOS diskettes are single density US format. The first few sectors are read at normal speed upon boot; the software determines whether the drive can handle UltraSpeed and then loads the high speed code into your computer. Even though double density sounds slightly slower than single density, the double density US format is even faster since it is working with larger sectors. Refer to the rest of the manual for more information about operation and formats.

If it doesn't work

Go over the instructions again and check your work. All of our products are thoroughly tested before shipping for high reliability. There is probably something you overlooked. If the U8 module is in backwards or not making a good connection, or if the jumpers are in the wrong position, the power light will come on but the drive will not spin. You can use the new U8 with your old U10 but not visa versa. If your drive won't boot the master DOS disk then try a standard boot disk of known quality. If you still can't get it to work, send your complete drive along with the MASTER SpartaDOS disk to us for repair.

Our service turn around time is generally 48 hours. If there is a problem with our parts there will be no charges. If there is a problem with your installation you will be charged a $25.00 flat rate including shipping. If there is a problem with the drive itself, our standard service rate is $40.00 plus parts and shipping. In any case we will send the repaired drive back to you via UPS COD. For repairs, send the drive and mark the box to:

ICD, Inc.
1220 Rock Street
Rockford, IL 61101-1437

Attn: 1050 Service

Warranty

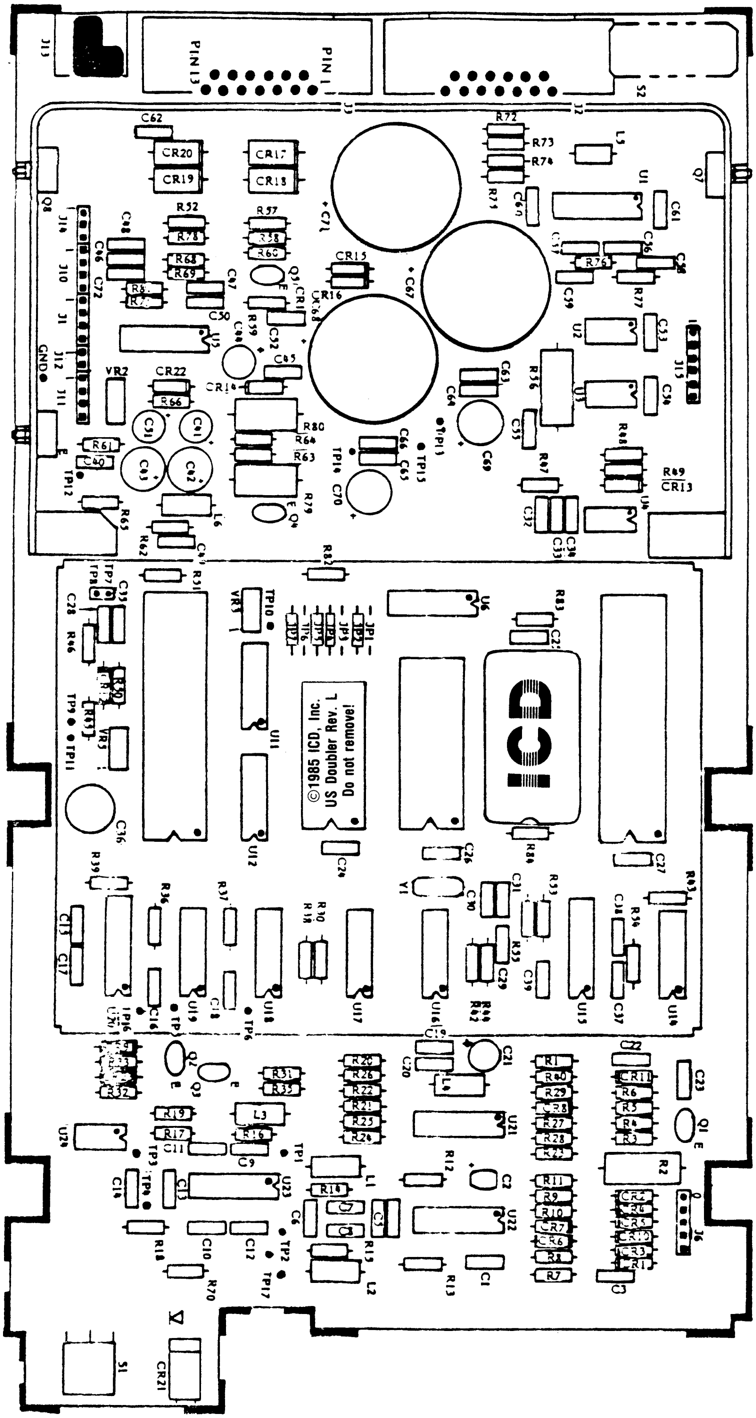
Be sure to completely fill out and return your warranty card. This is the only way you will be eligible for future updates or enhancements. The warranty is not transferrable and is intended for you as the end user only.

Warning! The warranty will be considered null and void if the copyright labels are removed from the ICs or if the hybrid module has been tampered with. We do **not** support 'pirates' (a nice word for theives dealing in computer software and hardware; we use other words). We own the copyrights for all of our products including SpartaDOS. Any users who are found to be selling or giving away copies of our products, forfeit all rights to any support or service. Furthermore, we **will** take legal action against those users if we feel it neccesary or justified.

SPECIAL CONSIDERATIONS

Format

Though the US Doubler is optimized for operation with SpartaDOS, any 'Atari Compatible' DOS should function with it properly. When changing from SpartaDOS to another brand of DOS and using the format command, first turn the drive power off, then back on (cold start) to reinitialize the internal format settings. Failure to do this could create format errors with the other DOS.



PIN 13
PIN 1
©1985 ICD, Inc.
US Doubler Rev. L
Do not remove!
ICD

APPENDIX G — US DOUBLER INTERFACE

The following is a list of the SIO commands of the US Doubler and their usage. For more information on how to perform Serial I/O operations you should consult the Atari OS manual.

Read Sector

Command:	R ($52)
AUX 1:	Sector number to read (low byte)
AUX 2:	Sector number to read (high byte)

Notes: The read (R) command operates exactly like any other Atari compatible disk drive. The data frame (sector size) received will depend upon the density/size of the diskette. The US Doubler automatically adjusts to a new sector size when a diskette is inserted into the drive. Sectors 1 through 3 will always be 128 bytes long. To determine the size, a status command must be executed, or (as in SpartaDOS), the size must be included somewhere within sectors 1 through 3.

Write Sector

Command:	Verify: W ($57) - No Verify: P ($50)
AUX 1:	Sector number to write (low byte)
AUX 2:	Sector number to write (high byte)

Notes: The write (W and P) commands operate exactly like any other Atari compatible disk drive. The data frame (sector size) sent is dependent upon the density/size of the diskette. The US Doubler automatically adjusts to a new sector size when a diskette is inserted into the drive. Sectors 1 through 3 will always be 128 bytes long. To determine the size, a status command must be executed, or (as in SpartaDOS), the size must be included somewhere within sectors 1 through 3.

Status

Command:	S ($53)
AUX 1:	— not used —
AUX 2:	— not used —

Notes: The status (S) command returns the status of the last operation (Controller status), the current operating status, and the approximate timeout value for a format command. A data frame of four bytes is returned by this command. They are as follows:

Byte 0: This is the controller status after the last command. A $FF indicates a good operation. (This is actually the 1's complement of the controller's status register - this is a bug propagated down from the old 810 drives.) The bits have the following meaning:

- Bit 0: BUSY — Should always be 1 (high).
- Bit 1: DRQ — Should always be 1 (high).
- Bit 2: LOST DATA — Should always be 1 (high).
- Bit 3: CRC ERROR — This indicates that there was an error in the last sector read if 0 (low). If combined with Bit 4 being low, the sector header exists but is unreadable (this means that the sector may not be written either).
- Bit 4: RECORD NOT FOUND — The sector does not exist if this bit is 0 (low).
- Bit 5: RECORD TYPE — If 0 (low), a special write command was given when the last sector was written. A normal drive (unmodified) will not create this type of sector. Note that the data is correct; it is a method of protection some publishers use.
- Bit 6: WRITE PROTECT — If 0 (low), the diskette was write protected. This should not happen since a write is never issued to the controller if protected. On reads, this bit is always 1.
- Bit 7: NOT READY — Indicates that the drive door is open if low (0).

Byte 1: This byte is the status the CPU generates indicating the following things:

- Bit 0: COMMAND FRAME — A 1 (high) indicates that the last command frame was in error. NOT USED BY THE US DOUBLER — always 0.
- Bit 1: CHECKSUM — A 1 (high) indicates that the last command/data frame checksum was in error. NOT USED BY THE US DOUBLER — always 0.

Bit 2: OPERATION — A 1 indicates the last operation was in error (bad sector, etc.) NOT USED BY THE US DOUBLER — always 0.
Bit 3: WRITE PROTECT — The diskette is CURRENTLY write protected if this bit is a 1.
Bit 4: MOTOR ON — The diskette is CURRENTLY spinning if this bit is a 1.
Bit 5: SIZE — The sector size is 256 bytes (in double density) if this bit is a 1.
Bit 6: — not used —
Bit 7: 1050 DD mode — This bit is 1 if in double density, but the sectors are 128 bytes long (‘DUAL DENSITY’).

Byte 2: This is the timeout value used when formatting by the computer's SIO routine.

Byte 3: — unused - always zero —

Format Diskette (General Format Command)

Command: ! ($21)
AUX 1: — not used —
AUX 2: — not used —

Notes: This command formats a diskette in either double or single density. (See the ‘N’ command for setting density.) A data frame of 128 bytes (if 128 byte sector format) or of 256 bytes (if 256 byte sector format) is returned. The US Doubler does not return the bad sector list. The first two bytes in the data frame will be $FF $FF.

Format Diskette (1050 Dual Density)

Command: ’ ($22)
AUX 1: — not used —
AUX 2: — not used —

Notes: This command formats a diskette in 1050 ‘dual’ density. (See the ‘N’ command for setting density.) A data frame of 128 bytes is returned. The US Doubler does not return the bad sector list. The first two bytes in the data frame will be $FF $FF.

Custom Format

Command:	f ($66)
AUX 1:	— not used —
AUX 2:	— not used —

Notes: This command formats a diskette in single, double, 'dual' density modes **and** allows the user to specify the sector ordering. The computer sends a data frame of 128 bytes to the disk drive. The first 12 bytes are the configuration bytes (as described under the 'O' command — set drive configuration), and the next 18 or 26 bytes are the sector numbers in order. The **standard** sequences are as follows:

single density:	17, 15, 13, 11, 9, 7, 5, 3, 1, 18, 16, 14, 12, 10, 8, 6, 4, 2.
double density:	18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1.
dual density:	1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26.

The standard **UltraSpeed** sector skews are as follows:

single density:	4, 8, 12, 16, 1, 5, 9, 13, 17, 2, 6, 10, 14, 18, 3, 7, 11, 15.
double density:	1, 14, 9, 4, 17, 12, 7, 2, 15, 10, 5, 18, 13, 8, 3, 16, 11, 6
dual density:	4, 8, 12, 16, 20, 24, 1, 5, 9, 13, 17, 21, 25, 2, 6, 10, 14, 18, 22, 26, 3, 7, 11, 15, 19, 23.

Return Configuration

Command:	N ($4E)
AUX 1:	— not used —
AUX 2:	— not used —

Notes: This command returns a 12 byte configuration table. This indicates the configuration the drive will format in next time a '!' format command is given. Refer to the 'O' (set drive configuration) command for definition of the 12 byte table.

Set Drive Configuration

Command:	O ($4F)
AUX 1:	— not used —
AUX 2:	— not used —

Notes: This command sets the configuration for the next format command ('!' command only). The computer sends the disk drive a 12 byte data frame which consists of the following (numbers are offsets within the data frame):

+0	Number of tracks (not used) - returns a 40
+1	Step rate (not used) - returns a 1
+2	Sectors/track high byte (not used) - returns a 0
+3	Sectors/track low byte - returns an 18 or 26
+4	Max head number (not used) - returns a 0
+5	Density - 0 if single, 4 if double density
+6	Bytes/sector high byte - 1 if 256, 0 if 128
+7	Bytes/sector low byte - 0 if 256, 128 if 128
+8	Drive present flag (not used) - returns 255
+9	— not used — returns a 0
+10	— not used — returns a 0
+11	— not used — returns a 0

Return High Speed Index

Command:	? ($3F)
AUX 1:	— not used —
AUX 2:	— not used —

Notes: This command returns a 1 byte speed index. This is the value which is used in the frequency register controlling the high speed SIO. The US Doubler currently returns a 10.

APPENDIX H — DISKETTES

Construction

The Atari disk drive uses 5 1/4 inch floppy diskettes. These are made of mylar with a magnetic coating and a dry surface lubricant much like recording tape. A semi protective lined jacket covers the diskette. Data is recorded in a digital format of 0s and 1s. Generally if a floppy disk loses just one of these bits your whole program is destroyed!

Storage and Handling

The general handling rules for diskettes are:

1. Never touch the actual recording surface of the disk. This is the shiny part on each side through an elongated hole.
2. Do not bend the diskette.
3. Keep diskettes in their protective jackets when not in use.
4. Do not expose them to extreme temperatures. If this happens let the disks sit at room temperature for at least 1 hour before use.
5. Keep the diskettes away from magnets or magnetic fields. This includes TV sets, motors, transformers and power supplies, etc.

Quality (TPI and Density)

Diskettes are rated by the guaranteed quality of the recording surface. Sometimes this is given in TPI or tracks per inch. A disk rated at 96 TPI and will have fewer errors than a 48 TPI disk. A disk rated for single density will probably have more errors than one rated for double density. The higher quality disks will probably last longer and create less wear on your disk drive head since they use better lubricants. Single sided disks will work with double sided drives but they are not guaranteed like the double sided variety.

Format Structure

Format on a single sided single density (SSSD) Atari drive consists of 40 tracks of 18 sectors each. Each sector holds 128 bytes but other DOS's use 3 of these bytes for mapping. Some of these sectors are also reserved for file management. SpartaDOS gives you 713 sectors of 128 bytes each for your use compared to 707 sectors of 125 bytes with Atari DOS 2.0. The raw SSSD format yields 92160 bytes per disk (90 KB).

1050 'Double Density' (1050DD) consists of 40 tracks of 28 sectors each. Each sector also has 128 bytes but is shorter in physical length. The raw 1050DD format yields 143360 bytes per disk (140 KB).

'True Double Density' (SSDD) uses 40 tracks of 18 sectors each but each sector stores 256 bytes. 184320 bytes (180 KB) is the total yield for a raw SSDD format.

Write Protect

There is a squared off notch on the upper right side of each diskette. This is the write protect notch. If this is covered with a protective tab, your drive will not be able to write to that diskette.

Note: The above holds true for 5 1/4 diskettes only. 8 inch disks are the opposite - cover the notch to write and uncover it to protect your data.

On Using Both Sides of the Diskette

A common practice seen mostly with home computers, is to cut a notch on the side of the diskette opposite the write protect notch, which can be done easily with a 1/4 inch paper punch. This allows the user to then flip the disk over and use the back side for storage which effectively doubles the amount of data that a single sided diskette can hold. This practice works with most Atari compatible drives but can lead to eventual diskette damage.

The problem begins when the disk is flipped to Side B; the drive spins the disk the opposite direction it was turning while on Side A. Before flipping, the felt fibers inside the jacket were always wiping one way keeping the disk surface clean and lubricated. Dirt accumulated on the bottom of the jacket as the fibers continuously cleaned the disk surface. When the disk is flipped to side B, the fibers bend the other direction and dump much of the dirt onto the top side of the spinning disk. The process is repeated when the disk is flipped back to side A.

We hope this will help you to understand the problems associated with using both sides of the disk. We don't discourage the practice because it has its uses, such as media distribution or for disks that are not used very often. However, we strongly recommend you only use the front side of a disk if you are using the diskette daily or when storing irreplaceable data!

APPENDIX I - GLOSSARY

The following is a list of terms and definitions which may appear throughout this or other computer manuals.

ADDRESS	a location in memory with the Atari from 0 to $FFFF
APPEND	to add on to. To append two files is to add one file onto the end of the other.
ASCII	the American Standard Code for Information Interchange which uses seven bits to define a one byte code from 0 to 127 decimal (0 - $7F). This standard was originated for early teletype machines and data communications.
ATASCII	a superset version of ASCII used only with the Atari computer. ATASCII defines a one byte (eight bit) code from 0 to 255 decimal (0 - $FF) and therefor can represent all possible codes on the Atari.
BANK	a predetermined size block of memory. Bank selecting refers to the practice of swapping different banks of memory in the same address space.
BATCH	a batch file is a file containing a group of commands to be executed consecutively.
BAUD RATE	the unit of speed for data transmission which is equal to the number of code elements per second. In practice it is often used interchangeably with 'bits per second'.
BINARY	the BASE 2 numbering system; only containing 2 numbers, either 1 or 0. For ease of use Binary numbers are usually represented in HEX notation. A Binary file is one which is directly readable by the computer without going through an interpreter.
BIT	a binary digit - either a 0 or 1
BOOTUP	refers to system initialization which sets up the computer when powering up. Also called BOOT.

BUFFER	any block of memory specifically set aside for use as temporary storage.
BYTE	the amount of information a computer can process in one cycle - the Atari byte •= 8 bits. The byte represents a number from 0 to 255 (0 to $FF HEX).
CIO	Central Input/Output. One part of the operating system that handles I/O.
COLD START	to start up the computer as if just powered up.
COMMAND	communication given from the human to the computer directing it to perform an action.
CP	Command Processor. The software interface between the keyboard handler and the DOS which allows the user to communicate with the DOS when entering a command line. When the command is entered, the CP will translate the command into information the DOS can understand and react upon accordingly.
CPU	Central Processing Unit. The intelligence of a computer system. The 6502 is the type of CPU used with 8 bit Atari computers.
CRC	Cyclic Redundancy Check. A method of data transfer error detection. As data bits are being transferred, they are manipulated mathematically to yield a highly sensitive error detection code that is appended to the data.
CURSOR	the pointer on the screen that marks where the next keystroke will appear or where the next action will take affect.
DATA	information generally used or operated on by a program.
DEBUG	to isolate and eliminate errors from a program.

DECIMAL	the BASE 10 numbering system; not very useful to computers unless translated to HEX or BINARY, but easy for humans to understand - uses the digits 0-9.
DEFAULT	the standard condition or value that exists upon running a program.
DENSITY	generally the number of bytes per sector is the disk density; single density being 128 bytes, double density being 256 bytes per sector. Actually density specifies the number of bytes per track on a disk.
DEVICE	the Atari defined devices are Dn:, E:, S:, R:, P:, C: ; referring to Disk drive (n=drive number), Editor portion of the screen display, the Screen, the RS232 device for communications, the Printer, and the Cassette storage device.
DIRECTORY	the list of all files stored in a given area of the diskette.
DOS	Disk Operating System ; this is the program which manages the I/O to and from the computer.
DMA	Direct Memory Access. DMA controls information flow directly into or out of memory without the intervention of the CPU. This causes data transfers to take place at a much greater speed than is possible with the CPU handling each byte of data.
DRIVER	same as HANDLER ; a program written to specifically handle one particular device or operation.
FILE	a collection of information usually stored as a named unit on a diskette.
FILESPEC	FILE SPECification. The information required in a command line to properly identify a particular file or group of files.

FORMAT	the guidelines for the way in which the magnetic structure of the disk is written ; standard Atari format is 40 tracks (complete circles around the disk) with 18 sectors per track (18 blocks of 128 bytes spaced around each track like pie pieces).
HANDLER	a program written to handle a device.
HARDCOPY	printed on paper.
HARDWARE	the computer, peripherals and their circuitry are hardware. The programming and documentation are generally called software.
HEADER	the first few bytes of a program which tell it where it should be located, what type of program it is and how it should be used.
HEX	the BASE 16 numbering system ; there are sixteen unique single digits used for counting 0-F. A HEX number is usually proceeded by '$'.
I/O	Input/Output ; this is what ties a computer to the outside world. This includes all devices (D:, E:, R:, P:, etc.) and peripherals.
ICD	Innovative Computer Design ; the company which wrote and designed SpartaDOS, the US Doubler, the R-Time 8, and other fine products for the Atari Computer.
IOCB	Input/Output Control Block. A 16 byte block of reserved memory which acts as a parameter passing window for I/O functions. There are 8 IOCBs numbered 0 - 7 although 0 is generally reserved for the screen editor (E:). One IOCB is needed for each OPEN device/file.
K	in the computer world, one K or kilo is equal to two to the tenth power or 1024.
KLUDGE	a 'Rube Goldberg' of software. A very complicated and confusing way of doing something relatively simple.

LANGUAGE	a program which makes it easier or faster in one way or another for humans to program a computer. BASIC, LOGO, PASCAL, Assembler are all languages for the Atari.
MACHINE CODE	the lowest level programming language but also the fastest running.
MODEM	MOdulator/DEModulator. A device which converts data from a form which is compatible with computers (bytes) to a form which is compatible with telephone systems (frequencies), and vice-versa.
NESTED	fitted within similar things.
PATH	the trail or course taken from one place to another ; when using subdirectories a path specifies the trail from where you are to where you want to go or retrieve a file from.
PARALLEL	the transfer, processing, or manipulation of all the bits in a byte simultaneously by using separate lines for each bit. Usually faster than serial which handles one bit at a time (sequentially) using a single line.
PERIPHERAL	an external device connected to your computer like a disk drive, printer, modem, etc.
PORT	a place of access to a system i.e. the serial communications port or the parallel joystick ports.
PROGRAM	a set of instructions to tell the computer how to accomplish some certain task. These instructions must conform to a particular order and the conventions of the language used.
PROMPT	a signal to the user that some action may be needed.
RAM	Random Access Memory. The computer can read and write to this but it is lost when power goes down.

REAL TIME	relating to 'real time' as on the standard clock ; a real time program uses a clock.
RELOCATABLE	a program that can be moved to different areas in memory and still function properly. Most SpartaDOS handlers are self-relocating. This means that when they are installed, they automatically relocate to MEMLO and then move MEMLO just above their code.
ROM	Read Only Memory. Permanent memory that can only be read.
RS-232	a communications interface standard designated by the Electronic Industries Association (EIA).
SECTOR	the standard block of storage used on floppy diskette media; can be 128 or 256 bytes with the Atari formats.
SERIAL	data transfer occurring on one signal line. The data bits are sent down the line sequentially.
SOFTWARE	the programming, documentation, and specified sequences of operation that allow a computer to function. Generally software refers to a program where hardware refers to circuitry.
SPARTA	a powerful city in ancient Greece ; POWER!
SYNTAX	the organization or arrangement of elements as parts of a command line.
TPI	Tracks Per Inch. This indicates how densely data can be packed on a diskette. TPI tells how many tracks cross a one inch segment of the radius of the diskette.
TRACK	a magnetic circle on the disk which contains the pattern of sectors. There are 40 tracks on a standard Atari formatted disk.
TRUNCATED	cut short.

VARIABLE	something that changes or has no fixed value.
WARM START	a SYSTEM RESET without wiping out memory as in cold start.
WILD CARD	used when specifying filenames or pathnames to ease operator entry or select a certain range of names. * and ? are the two valid wild cards.
WORD	an ordered set of characters which occupy one memory location. Generally an 8 bit computer word is 8 bits (1 byte), a 16 bit machine has a 16 bit word.
XIO	a general Input Output statement used in a program for Disk I/O and in Graphics work.

IMPORTANT WARRANTY INFORMATION
LIMITED 30 DAY WARRANTY

ICD, INC. warrants to the original consumer purchaser that this **ICD, Inc.** Personal Computer Product (not including computer programs) shall be free from any defects in material or workmanship for a period of 30 days from the date of purchase. If any such defect is discovered within the warranty period, **ICD, Inc.'s** sole obligation will be to repair or replace, at its election, the Computer Product free of charge on receipt of the unit (charges prepaid, if mailed or shipped) with proof of date of purchase satisfactory to **ICD, Inc.**

Write to:

ICD, Inc.
1220 Rock Street, Suite 310
Rockford, IL 61101-1437
Attn: Service Dept.

YOU MUST RETURN DEFECTIVE COMPUTER PRODUCT FOR IN-WARRANTY REPAIR.

This warranty shall not apply if the Computer Product: (i) has been misused or shows signs of excessive wear, (ii) has been damaged by improper installation, or (iii) has been damaged by being serviced or modified.

ANY APPLICABLE IMPLIED WARRANTIES, INCLUDING WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE HEREBY LIMITED TO THIRTY DAYS FROM THE DATE OF PURCHASE. CONSEQUENTIAL OR INCIDENTAL DAMAGES RESULTING FROM A BREACH OF ANY APPLICABLE EXPRESS OR IMPLIED WARRANTIES ARE HEREBY EXCLUDED. Some states do not allow limitations on how long an implied warranty lasts or do not allow the exclusion or limitation of incidental or consequential damages, so the above limitations or exclusions may not apply to you.

This warranty gives you specific legal rights and you may also have other rights which vary from state to state.

DISCLAIMER OF WARRANTY ON ICD, INC. COMPUTER PROGRAMS: All **ICD, INC.** computer programs are distributed on an "as is" basis without warranty of any kind. The entire risk as to the quality and performance of such programs is with the purchase. Should the programs prove defective following their purchase, the purchaser and not the manufacturer, distributor, or retailer assumes the entire cost of all necessary servicing or repair.

ICD, Inc. shall have no liability or responsibility to a purchaser, customer, or any other person or entity with respect to any liability, loss, or damage caused directly or indirectly by computer programs sold by **ICD, Inc.** This disclaimer includes but is not limited to any interruption of service, loss of business or anticipatory profits or consequential damages resulting from the use or operation of such computer programs.

REPAIR SERVICE: If your **ICD, Inc.** Personal Computer Product requires repair other than under warranty, please write to **ICD, Inc.**, Service Department for repair information.

IMPORTANT: If you ship your **ICD, Inc.** Personal Computer Product, package it securely and ship it, charges prepaid and insured, by parcel post or United Parcel Service.

WARRANTY/UPDATE REGISTRATION CARD

Please take the time to complete this card and return it to us to allow us to provide you with more efficient service, including updates, should your **ICD, Inc.** product require it.

(Please print)

Name____

Address______________________________________

City____________________ State____________________

Country__________________ ZIP____________________

Phone (   ) __________ Item Purchased US DOUBLER
(Area Code)

Date of Purchase__________ Serial Number 039166

Where Purchased________________________________

What other products would you like to see us develop?____________

Does your local Atari dealer carry our product line? ☐ Yes ☐ No

Your Atari dealer's name, address________________________

PLACE

STAMP

HERE

ICD, Inc.
1220 Rock Street, Suite 310
Rockford, Illinois 61101-1437