THE COMPLETE
SPARTADOS CONSTRUCTION SET
MANUAL

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 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 (enhanced) 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 culmination of two major versions and several SpartaDOS types with many powerful utilities. This provides you, the user, with the building blocks for creating your own DOS disks. 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 disks. 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.

l

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 . . .

- o manages the allocation and de-allocation of files
- o provides a set of commands to interact with it
- o provides a means of parameter passing to the user programs
- o provides a set of useful tools to aid in software development
- o oversees the allocation of memory
- o 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 one), 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 computers 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 drive. It is on a disk and then read into the computers 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 drives will not be recognized by the system. Always turn drive 1 on before the computer, insert your DOS disk into the drive and then power up the computer. The computers 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 disks out of the drive before turning the power off. Failure to do this may write bad information on the disks when using 810 drives, 1050 drives are OK for this.

Different uses of 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 and date stamping have become invaluable in a well organized filing system. SpartaDOS is the only DOS that allows time and date file stamping on the 8 bit Atari computer. Subdirectories, like file folders, allow you to save different files under different categories. Time and 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.

BINARY FILE LOADER
Binary files are machine language programs in file form. 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 information 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

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

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

o 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).

o 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.

o 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-F to represent decimal values of 10-15. 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.

o Many commands have restrictions as to what DOS and what format on disks 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:

General terms used

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

Atari DOS 2 disks
This refers to any disk 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 disk, since there are no subdirectories on these disks. 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 disks 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 one to eight characters in length.

.ext This is the filename extension, thus it represents from 0-3 chars.

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 disks.

Volume Names
When formatted, SpartaDOS disks are given a volume name. Each disk should be given a unique volume name such as Games_1, Games_2, WP_1, 000243 etc. Volume names can be from 1-8 characters long and may include any of the 256 possible numbers, characters or symbols, available on the Atari keyboard.

Version 1.x disk 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 disks 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 disk. If the old and the new disks have the same volume name, SpartaDOS will not know there is a new disk in the drive and consequently the new disk is in danger of being updated with bad information. Even though version 2.x disks have extra protection (random and sequence numbers), if they are used under version 1.x SpartaDOS, the extra protection is not used.

Directories

The disk is broken up into directories, which may contain up to 127 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 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 disk being placed in the 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 more 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 placeholder 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 2 menu programs. One is a binary loader (discussed in chapter 10) and the other is a general command menu (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 disks. 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. You cannot use double density format with the standard 810 drive or double sided format with any 810 or 1050 drive. These options may seem to format an incompatible drive with no errors, but the disk will not be fully functional. Once a format is written on a given disk, its drive will automatically configure for that format type when trying to read or write to it. See chapter 5 for more on formatting.

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 1 of 2 categories: 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 characters)

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

```
 TEST1.123-Legal
 A FILE.LST-Illegal-no spaces allowed
 ANOTHER.FIL-Legal
 4TH.TRY-Legal name
 B_FILE.JN-Legal
 PROG.BASIC-legal (IC part of. BASIC won't be known)
 DATA#-Illegal character
 B-Legal
```

Actually, any filename is legal under SpartaDOS version 2.x, but once an offending character is encountered, no other characters are accepted. Thus, the 2nd 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 (load and go file)
.BAS-BASIC saved program
.TXT-ASCII text file
.OBJ-An object code file
.SYS-A system file
.EXE-An executable file
.ASM-Machine language source file listing
```

.BIN-Binary data file
.DAT-general data file
.PRN-listing to be printed
.BAT-batch file
.HEX-hexadecimal coded file
.FNT-a font file
.LST-LISTed Basic file
.SRC-general source file
.MUS-Music file
.DOS-SpartaDOS module (file) for INIT/XINIT commands

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 ? is a don't care character. This means that it will match any character in its position. The * is like a repeat until period or a repeat until end of filename question mark. An * 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 cards:

*.BAS	Represents the files that have an extension of 'BAS'
.	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 as a 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 then 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 (towards 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  3:39p
    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    path=   <MODEM>RS232.COM or
                     >MODEM>RS232.COM

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

TESTPROG.BAS path=   <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    path=   <<MODEM>RS232.COM or
```

```
                    >MODEM>RS232.COM

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

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

Note that the 'path' 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 (ie. 'D2:') followed by a RETURN. The 'D' and the colon (:) ARE REQUIRED. In the following example, the user input is in caps.

```
   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 and 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:

```
D1:DIR

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):

```
D1:DIRS

 *AT-RS232 COM 016
 *CHTD     COM 009
 *UTIL     DIR 002  DIR part is in 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

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.

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 disk onto the newly formatted disk. Re-insert the disk 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 disk, remove the master disk and replace it with the one you just formatted. Now press RETURN again. If asked to insert the source disk, replace the newly formatted disk with the master disk 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 disk away so that it will not be damaged.

Overview of Common 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> and then 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 disk 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 one with a RETURN):

```
RENAME TEST.   *RANDOM.*
ERASE RANDOM.DAT
CAR        (NOW IN BASIC)
LIST       (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:

DIR and 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 disk. The DIRS command lists the directory in a DOS 2 (CP version 2.x only) kind of way. 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-2.x
DIRS is internal under CP version 2.x

Remarks

If no file spec is specified, all files will be listed (ie. 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 1.x. With version 2.x, DIRS displays a short form similar to Atari DOS 2 including the protected status (which DIR doesn't 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 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 (lockup) 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.

The format disk command was very simple when the Atari 810 was the only drive available. The only choice was single density, single sided and 40 tracks, 1 command was sufficient. As 3rd party vendors developed more sophisticated drives, Atari owners suddenly had a choice. Percom, a leader in new products introduced double density drives, and 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 and XINIT Commands
Purpose - These are the master formatting programs for SpartaDOS.

Syntax
INIT
XINIT

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

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 disk. The FORMAT command is a stripped down version of the INIT command which reads the DOS from an already existing version 1.x disk 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 disk but want it formatted. Assuming you selected a DOS, the correct DOS module (file) 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 disk is booted. Next you will be asked which drive you want to format, valid selections are from 1-4 with INIT and from 1-8 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 enhanced density (128 byte sectors).

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

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 disk to be formatted and press RETURN. Disk initialized...will appear when finished. To format more disks, press RETURN, to leave this program, press the ESC key.

There are currently 4 versions of SpartaDOS 1.x and 2 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. 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 (1 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 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 drive and are not using an XL/XE computer.

SPEED.DOS

This is STANDARD 1.x version with the UltraSpeed code added. Use this version of your regular DOS if you have the US Doubler in a 1050 drive and don't have an XL/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, therefore 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 disks. 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 DOS or the Cartridge after 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 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 also gives you the complete Command Processor as in the XD version.

AINIT Command
Purpose - 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 tech 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 disk formatted in this way either. Use INIT for SpartaDOS 1.x or XINIT for SpartaDOS 2.x disks.

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 disk, create the directory structure and optionally put DOS on the disk.

Syntax
FORMAT

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

Remarks
The FORMAT program allows many format densities, gives the user the option to put DOS on the disk and to give the disk a unique volume name. Once a disk has been formatted, DOS cannot be put on the disk 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 disk 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 disk 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 disk 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 is valid)
Select number of tracks (usually #1)
```



```
Select Density?
Volume name (1-8 chars)
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 3 sectors of each SpartaDOS 2.x formatted disk, 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 disk, first copy the new DOS file to the disk, then use the BOOT command to force the new DOS to execute upon system boot. To create a binary boot disk, 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 disk is booted, it will immediately try to load and run the boot up file STAR.BIN

RAMDISK Commands
With the introduction of the 130XE 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 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 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 1-8 is valid with CP version 2.x, drives 1-4 are valid with 1.x. If there is already a drive in the selected location, the 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 eight 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: MEMU.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 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 the 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 version 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 disk 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 disk 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 are 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.

THIS COMMAND WILL NOT COPY A FILE BETWEEN TWO DISKS USING THE SAME DISK DRIVE. COPY will copy files to the same disk, however, the new file must have a different name or the destination directory must be different than the source. There is no provision to switch disks 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 2 files (adding one file to the end of another).

You may also use COPY to transfer data between any of the other system devices, ie: the screen editor, printer, keyboard, etc.

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 filename. 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 disk 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 line in the command line. The address header(s) from the first file will also be copied onto the end of the second 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 from 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 disks.

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 disk 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 disk as necessary.

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

SYSTEM I/O ERROR: This is a general purpose error message given by SPCOPY when something goes wrong, ie. inserting the wrong disk when swapping source and destination, copying between 2 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 than SCOPY 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 disk if necessary.

2. The file tagging has been improved so that you can see 4 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 disks in particular). PLEASE GIVE ALL YOUR DISKS DIFFERENT VOLUME NAMES!

DUPDSK
Purpose - To duplicate an entire SpartaDOS disk (except for volume name), using

one or two drives. IMPORTANT: THE NUMBER OF TRACKS AND THE DENSITY ON THE SOURCE AND DESTINATION DISKS 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 disk including subdirectories while using one or two drives. This command WILL NOT FORMAT OR TRANSFER THE DISK 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 disk 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 disk?' if a single drive copy or to 'Insert Source and Dest. Disks?' if a two drive copy. Press any key (except ESC) to start the duplication and repeat as necessary if swapping disk 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 disk 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 WARNINGS will be given. Note that only files will be erased, any subdirectories will be left intact. To restore erased files, see the UNERASE command. The UNERASE

command MUST be used before any more data is written to that disk or you may lose your data!

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 likely has been overwritten and therefore lost forever!

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 using 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 can 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 Chapter 4.

Example
  RENAME FILE FILES

This command changes the name of the file on the default drive and default directory from FILE 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 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 the 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 byte. If set, that file may not be modified in any way. The following give 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 versions 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 or 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 disk 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 versions 2.x
SpartaDOS 1.x version 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 the 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

Creating a binary 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 or 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 and run type of menu that you can't exit aside from rebooting, only reading from the disk is performed.

b) UltraSpeed - Both versions of DOS run in UltraSpeed mode as long as your drive hardware supports it and you select UltraSpeed (US Doubler) 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 doublewide 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 and OPTION keys as described in (c). Once a disk 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 categories (subdirectories) on one disk, but only allow certain categories 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 disk.

Construction for non XL/XE computers

Initialize a disk with NOCP.DOS using INIT.

This is a version 1.x type SpartaDOS. Insert a disk 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 one, tracks will be one 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) enhanced (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 disk. You might want to use numbers or letters or both, as it is intended to help you keep track of your disk. US Doubler owners will type Y (yes) for UltraSpeed sector skew, everybody else will type N (no).

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

COPY LOGOMENU.SYS TO THE MAIN DIRECTORY ON EACH OF YOUR NEWLY INITIALIZED DISKS AND RENAME IT TO AUTORUN.SYS

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

CREATE SUBDIRECTORIES FOR FILE STORAGE ON EACH OF THE DESTINATION DISKS. 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 (ie. SPACE for space games, MAZE for maze games etc). Example: CREDIR MAZE. This writes a subdirectory called MAZE on the disk under the MAIN directory.

COPY THE FILES TO YOUR NEW DISKS 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: MAZE>*.* This will only work if the subdirectory named MAZE 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 disk. RESET reloads the directory. This is helpful in the event that you change disks 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 property of Atari Corp.

TROUBLE SHOOTING:
The display comes up with READ or MEMO PAD/DIAGNOSTICS - NOCP.DOS needs a file called AUTORUN.SYS in order to initialize properly. Check MAIN directory for 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 disk.

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 MAZE> will show a subdirectory called MAZE.

Construction for XL/XE computer ONLY

INITIALIZE A DISK WITH XC23B.DOS USING XINIT.

Insert a disk with XINIT and the 2.x versions of DOS into drive one 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 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 2s 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) 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 ram and use it from the menu program. If

'n' is not specified, drive one is assumed. Note: if you specify both 'R' and an 'n', do not put a space between them (ie. R5). The operation of MENU follows in the next section

The MENU Operation

Many of the MENU Functions (ie. 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 keystrokes 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 keystrokes 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 one.

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 one. Next enter the path name of the destination directory or RETURN for the MAIN directory. When prompted to insert disks, 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 again.

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 disks. To exit the XINIT program and re-enter MENU, press the ESCape key.

AInit (A) This command formats a disk in Atari DOS 2 format. Enter the drive number when asked (RETURN for drive one) 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 disk in the current drive.

UnLock (O) This command write unlocks the disk 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 (ie. D2:). The screen is cleared before the copy. After it's 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 insure 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 and 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 and 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 data 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 Sparta; 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, and then move MEMLO just above their program area.

TO SET TIME and 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, 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 8 (TD) until the computer is turned off. For instructions on how to set the R-Time 8 or the software clock, refer to the SET and TSET command descriptions that follow in this chapter.

Note: The R-Time 8 cartridge is a very accurate, crystal based timing device which works on both 60Hz and 50Hz 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 data display off. Similar to the TD command but for use without the R-Time 8.

Syntax
TIME [X]

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

Remarks
If only TIME is entered, the time and 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 reinitialized with every RESET. The time and 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 with battery backup, the TSET command must be used.

Example
  SET

Since no parameters were specified, the prompt showing the current data and asking for a new date appears. Type in the new data 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 and 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 9:12xxpm. 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 8 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 8 60 times a second and update the display every second. If the R-Time 8 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 and date in the R-Time 8 use the TSET command.

The R-Time 8 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 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 into the initialization vector and is re-initialized with every RESET. The time and 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 to load the time and date into the system when you do not want the time and date display.

Syntax
XTD

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

Remarks
If XTD is entered and the R-Time 8 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 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 and date in the R-Time 8, use the TSET command.

NOTE: This command patches itself in the initialization vector and is re-initialized with every RESET. The time and 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 data 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 and 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 files time and 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 and date stamp on a file. This can be useful for files which come from a DOS disk which doesn't support time and 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 and 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 (ie. 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 BBS's 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 sent 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, 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 (ie. 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 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 computers 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 (ie. 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, Aux 1 and the second byte is for XIO 38, Aux 1. 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 FL 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 disk. Its 2 bytes set the port at 4800 baud, no parity, 8 data bits, 1 stop bit and send LF after CR on output.

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 batch 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 disk 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 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 max 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 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 disk. For instance you may want the R-Time 8 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:

Example
```
  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, ie. 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 (ie. 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 file called 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 RESET.

Consider execution of the following batch file from drive 1:

Example
```
  RS232
  ;Please insert your communications
  ;program disk 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 1 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 or 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 or device is closed, and output is echoed to the new file or device. If no parameter is given after the PRINT command, the file or 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, ie. BASIC when entered, closes all IOCBs.

Example
```
  PRINT P:
```

This command sends all future screen display to the printer until another PRINT command 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 (HATABS). This is a list of device letters (ie. E K S P D...) followed by a handler table address. Sparta 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 char, Put char, 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 Systems (OS) handler(s) 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. 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

Why a 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 has 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 (ie. 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___INFORMATION 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 ROMs). 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 file takes. MEM will also give you an idea of the free memory available. You can see if the internal BASIC (XL/XE) is installed noting the MEMHI location. The MEMLO command only displays MEMLO.

Example
  MEM

This command displays the MEMLO and MEMHI values in this 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 2-16. 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 DOSes. This default can be changed when formatting a new version 1.x disk 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 two, and for double density read and write is four. 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 drives.

Sparta 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 drives speed and to determine the amount of free space on your disk. 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 and sequence numbers (version 2.x disk only), sector size, formatted bytes on disk, available bytes on disk and write lock status (version 2.x disks only.)

Syntax
CHKDSK [Dn:]

Type and Restrictions
Internal under CP version 2.x

Remarks
This command is used for determining information about a disk. If the disk was initialized with XINIT (version 2 format), CHKDSK will give a volume name then random and 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 per 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 disk. The 'Write Lock' is a software write protect ('ON' indicates locked or protected).

If the disk is NOT a version 2 format, the random and sequence numbers and write lock status are omitted from the display. If the disk 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 OA 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 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 disk 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 passed and used).

Command Files
Purpose - To load and run binary files. 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

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, and if the file has no extension, be sure to end the name with a period (.).

Command files are run at the beginning of the location of their first segment load after they are fully loaded. Run and INIT addresses ($2E0 and $2E2) are also supported with the RUN address ($2E0) having priority over the run at first location loaded 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 because of its period at the end.

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 file at memory location $4000.

```
  RUN
```

This command runs the last file executed. If SCOPY 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 files.

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 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 and end addresses, 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 $8FFF 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.

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 disk.

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 8 memory locations across the screen 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 with 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 8 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 address, file position for beginning of segment and can query whether to load a given segment. It 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 disk.

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-$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 (Query) stops before it loads each segment and asks 'Load this segment?' Answer with Y/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 /SN
```

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 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, any misconceptions you may have will be cleared up.

Basic Operation WITHIN the Drive.

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

1) The Computer and Disk Drive Interface - This is commonly referred to as the SIO (Serial Input and 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/cable lines.

b) The computer sends a command frame. This consists of 4 bytes. They are 1) device ID--each unit on the serial port has a unique device ID, 2) command--such as read/write/status/format, 3) and 4) two bytes of auxiliary information--such as the sector number in 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 (drive) identified by the device ID, answers by sending an ACK ($41(A), if the command is valid) or a NACK ($4E(N), if its 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 ($43(C)) or an ERROR ($45(E)) 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 drive called a controller. This device manages the specifics of the disk 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. 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 one buffer in single density or two 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 (ie. DUPDSK, INIT, XINIT, etc) use this vector so

they can take advantage of the high speed I/O. For more information on this vector (LSIO), 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 being 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 ($3F, 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 drive after it has written the sector. It is often believed than 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 the normal speed write (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 disks, they are optimized to allow to sequential sectors to either be read or written in one revolution. Thus 10 sectors maybe read in 5 revolutions (which is about one second). With verify on, it takes 1.5 revolutions to write a sector (which is about three seconds to write 10 sectors). The US Doubler optimized 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 one 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 and Restrictions

Internal under CP version 2.x

Remarks
Verify means to write a sector, read it back internally in the drive and compare it with drive 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 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 are having drive problems.

Note: When you format a version 1.x disk, you may select 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 easy and then steps quickly into the machine language world.

SpartaDOS Functions from Basic

The following is a list of SpartaDOS function 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 1-7. Atari disks refers to Atari DOS 2 type formatted disks.

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 are reading an Atari DOS 2 disk. 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 and 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 DISKS.

24 Open the current directory in update mode. This is the raw data of the directory and can be read or written like any other file. THIS MODE IS ONLY AVAILABLE ON SpartaDOS FORMATTED DISKS.

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 DISKS.

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 have a 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 disk. Atari DISKS 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 disk. Atari DISKS MAY ONLY BE ACCESSED WITH A VERSION 2 SpartaDOS LOADED INTO THE COMPUTER.

Lock Disk (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 DISKS 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 disk. 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 maybe used in the filenames. This function is valid for any format disk. 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 DISKS: In the first method, position (POS) MUST be from 0-32767. The second method may take position up to 8,388,607 ($7FFFFF in Hex notation). You may position beyond the end of file if the file is opened in read and 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 DISKS: 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 disks. This works identically like Atari DOS 2. Atari DISKS 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 DISKS: In the first method, position (POS) WILL be from 0-32767. 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 DISKS: 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 disks. This works the same as Atari DOS 2. Atari DISKS 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 ONLY WORKS FOR SpartaDOS formatted disks. Atari formatted disks 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 and RUN vectors will be used, otherwise they will be ignored. Note that the IOCB must not be open. Atari DISKS 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 over written. 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 DISKS 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 DISKS.

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 DISKS.

Change Working Directory (CWD)

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

Notes
This changes the current default directory. The pathname must be valid and all directory names in the path must exist. Note that the IOCB must be closed. THIS WILL NOT WORK ON Atari DOS 2 DISKS.

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 maybe 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 Disk (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 Disk 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 LOADED INTO THE COMPUTER.

Directory Listing (DIR)

Syntax
```
10DIM 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 12, 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 CAN ONLY BE ACCESSED WITH A VERSION 2 SpartaDOS LOADED INTO THE COMPUTER.

COMTAB EQUATES

Because SpartaDOS is mainly a Command Processor driven DOS, a great number of variables have been made user accessible. These are mainly for parameters passing on the command line, time and 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 memory 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 IDs 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 to handler IDs 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, 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 and 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 Disks

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

Boot Sectors

The boot sectors are the first 3 sectors on all SpartaDOS disks. 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 disk.
13-This is the number of free sectors on the disk.
15-This is the number of bit map sectors used on the disk.
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 disk volume name (8 chars). These must be unique on SpartaDOS version 1.x disks OR when using a version 1.x SpartaDOS.

30-This is the number of tracks the disk has. The most significant bit is set if this is a double sided drive.
31-This is the size of the sectors on this disk. 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 $10 (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 DISKS.
34-This is the default drive the Command Processor uses if this disk is booted.
35-RESERVED
36-RESERVED
37-This is the number of sectors in the main DOS boot (UNDER VERSION 1.x DISKS 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 disk has been changed. ONLY APPLICABLE TO SPARTADOS 2.x DISKS.
39-This is the volume random number. This is simply a random number generated when the disk was formatted. Its function is the same as the volume sequence number. ONLY APPLICABLE TO SPARTADOS 2.x DISKS.
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 2.x DISKS.
42-This is the write lock flag. A value of $FF indicates the disk is locked, and a 0 indicates that it is not. ONLY APPLICABLE TO SPARTADOS 2.x DISKS.

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 8th sector in the group. The first byte of the bit map corresponds to sector numbers 0-7, the second corresponds to sector numbers 8-15, etc. (NOTE that sector number 0 does not exist). If more than 1 bit map is required for

the disk, they will be sequential on the disk. A sector is free if the corresponding bit map byte is SET (1).

Sector Maps

Sector maps are simply a list of sectors allocated to a file or a directory. The first two entries in a sector (allocation) map 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) sector numbers making up the file or directory entries.

next: This is the sector number of the next sector map. It will be a 0 if

this is the last sector map.

last: This is the sector number of the last (previous) sector map. It will be 0 if it is the first sector map.

data: These are the sector numbers of the data sectors of the file. If a data sector number is 0, 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).

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 and 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 0 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:
bit 0-The entry is protected.
bit 3-The entry is in use.
bit 4-The entry has been deleted.
bit 5-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 data 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 Disk Status (CHKDSK)

This function retrieves information about the disk. 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 disk: 0 if Atari DOS 2
     +1    = number of bytes per sector: 0 implies 256
     +2    = total sectors on disk (low,high)
     +4    = free sectors on disk (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/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 the 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 are:

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 disks automatically. If an Atari DOS 2 disk is in the 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 as in the SpartaDOS handler with the exception of NOTE and POINT. These preserve the Atari DOS 2 interpretation (section #/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 disks.

   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 (ie. ERASE, CREDIR etc).

7) Batch files may now be linked (ie. 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 and 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 disk. 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 and UNPROTECT commands have been added. They work identically to the Atari DOS 2 commands. The XIO codes are the same as described by Atari DOS 2. (protect=35, unprotect=36)

11) Disks may be software LOCKed and UNLOCKed. If a disk is locked, it will act just as though a write protect tab is on the disk, however a different error code is returned. The XIO codes are: lock=34, unlock=46.

12) The following error codes and meanings have been added:
 $95=Not version 2.x disk
 $94=Not a SpartaDOS disk
 $A3=Illegal wild card in filename
 $A4=File protected (attempt to replace was made)
 $A9=Disk is write locked

13) The VERIFY command has been added. This allows the user to change between write with verify and write without verify.

14) 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.

15) 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 0 filled before.

16) An error will be given if SpartaDOS 2.x is read into a non-XL/XE 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 disk.

17) The BOOT command has been added. This command will select a program (normally DOS) to load when the disk is booted. The file selected must

be a standard binary load file. The INIT and RUN vectors are handled normally (as described under Atari DOS 2 and SpartaDOS). This is the XIO function number 45.

18) The DOS loader (on the first 3 sectors of each disk) 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 file must not overwrite these areas.

19) 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.

20) 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)

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

22) A CHKDSK command has been added. It states the volume name, the random and sequence numbers, the total bytes per disk, the free bytes per disk, the sector size and the write lock status. Only SpartaDOS 2.x disks will display the write lock status and the sequence and random numbers. This is also an XIO function (described in chapter 19).

23) 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 (ie. XD23B.DOS etc).
    e. at +42 in sector 1 is the write lock flag, $FF is locked.

24) The AINIT command has been added. This is the Atari DOS 2 format command, and is also XIO function number 254. A Yes/No prompt is given

for safety.

25) The following are new definitions of offsets within COMTAB (numbers represent offsets):
 -2 SIO command used to write a sector (P or W)
 -7 Flag indicating active DIVIN ($FF is false)
 -8 Flag indicating active DIVOUT ($FF is false)

26) 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.

27) High Speed I/O routines are placed under the OS ROM. During boot up of the disk, 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 greater than 32767
8	Input statement error
9	Array or string DIM error
11	Floating point overflow or 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 zero or greater than 7
21	LOAD file error

SpartaDOS Error Messages

Code #	Error Code Messages
128($80)	BREAK abort
129($81)	IOCB Already open

130($82).Nonexistent device specified
131($83).File's IOCB not open for read
132($84).Invalid IOCB command
133($85).Device or file's IOCB not open
134($86).Bad IOCB number
135($87).File's 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 or disk write protected)
146($92).Function not implemented in handler.
148($94).Not a SpartaDOS disk
149($95).Disk not SpartaDOS 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 - illegal characters in filename
166($A6).Position range error
167($A7).Cannot delete directory
168($A8).Illegal DOS command is not possible
169($A9).Disk is write locked
170($AA).File not found - misspelled command, path>fname or a write operation was tried with NOWRITE.DOS

APPENDIX B___COMMAND SUMMARY

?DIR [Dn:][path>] pg22
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:] pg18
Internal-2.x This command is used to format a disk in Atari DOS 2 style format.

APPEND [Dn:][path>]fname[.ext] address address pg58
Internal-1.x and 2.x This command saves a binary block of data at the end of an existing binary file.

AT_RS232 pg45
External-1.x and 2.x Loads the RS232 handler for the ATR8000.

AUTOBAT [Dn:][path>]fname.ext pg127
External-3.2+ ONLY Causes specified batch file to be run whenever RESET is pressed. For SpartaDOS 3.2 and above.

BASIC ON or BASIC OFF pg15
Internal-2.x This command installs or removes the internal BASIC with the XL/XE computers.

Batch Files (syntax below)
-fname pg47
Internal-1.x and 2.x To retrieve and execute a batch 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 disk is booted.

BOOT [Dn:][path>]fname[.ext] pg20
Internal-2.x This command tells a SpartaDOS 2.x formatted disk to boot a particular program at startup.

BUFS [n] pg53
Internal-1.x To set or check the number of buffers currently in use

under CP version 1.x only.

BYPASS pg126
External Causes Supra Hard Disk Interface to use floppy drive one instead of hard disk drive one.

CAR pg15
Internal-1.x and 2.x Exits from DOS to a language cartridge.

CHKDSK [Dn:] pg54
Internal-2.x To display the volume name, random and sequence numbers (version 2.x disks only), sector size, formatted bytes on disk, available bytes on disk and write lock status (version 2.x disks only).

CHTD [Dn:][path>]fname[.ext] pg43
External-1.x and 2.x This utility command is used to change a files time and date stamp.

CHVOL [Dn:]vname pg31
External-1.x and 2.x This utility command is used to change the volume name on a disk.

CLEANUP Dn: [/P] pg112
External-1.x and 2.x This utility command detects and corrects file structure defects on SpartaDOS disks. Option P sends optional report to printer.

Command Files (syntax below)
[Dn:][path>]fname [parameters] pg55
Internal-1.x and 2.x To load and run binary files. It also provides a standard for passing parameters to machine language programs.

COPY d[n]:[path>][fname[.ext]] [dn:][path>][fname[.ext]] [/A] pg25
Internal-1.x with exceptions and 2.x Note: The /A option is for CP version 2.x

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 file 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 disk. There is no provision to switch disks 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, ie: Screen Editor, Printer, Keyboard etc.

CREDIR [Dn:]path pg22
Internal-1.x and 2.x Creates a subdirectory on the specified disk.

CWD [Dn:]path pg23
Internal-1.x and 2.x Change the working directory on the disk.

DATE pg121
Internal-3.2+ ONLY To display and set date for SpartaDOS 3.2 and above.

DELDIR [Dn:][path>][fname[.ext]] pg23
Internal-1.x and 2.x Deletes a subdirectory entry. You must first delete all files in a subdirectory in order for DELDIR to work.

DIR [Dn:][path>][fname[.ext]] or
DIRS [Dn:][path>][fname[.ext]] (optional with CP version 2.x) pg14
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 create and the number of free sectors left on disk. DIR may be used to list all files matching a filespec 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 pg51
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.

DISKRX or
DISKRX Dn: or
DISKRX [Dn:][Path>]fname.ext pg107
External-1.x and 2.x This utility command edits sectors, traces files and rebuilds directories etc.

DOSMENU pg103
External-1.x and 2.x This utility command replaces Command Processor with an Atari DOS 2 type menu driven DOS.

DUMP [Dn:][path>]fname[.ext] [start [#bytes]] [/P] pg58
External-1.x and 2.x This utility will display a file or portion of a file in HEX/ATASCII/ASCII format.

DUPDSK pg28
External-1.x and 2.x To duplicate an entire SpartaDOS disk (except for volume name) using 1 or 2 drives. Note: Number of tracks and densities must match on source and destination disks or an error will result.

ERASE [Dn:][path>][fname[.ext]] pg29
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 pg19
External-1.x and 2.x This command is used to format the disk, create the directory structure and optionally put DOS on the disk. Only creates 1.x disks. Use XINIT for 2.x disks.

INIT pg16
External-1.x and 2.x This is the master formatting program for

SpartaDOS 1.x versions and allows selection of certain default parameters. Only creates 1.x disks. Use XINIT for 2.x disks.

KEY or pg52
KEY ON and KEY OFF pg123
External-1.x, Internal-3.2+ To install a 32 character keyboard buffer for 400/800 computers. Use XKEY for 2.x SpartaDOS. Use KEY ON/OFF for SpartaDOS 3.2 and above.

LOAD [Dn:][path>]fname[.ext] pg56
    Internal-1.x and 2.x This command loads any binary file into memory but does not run the file. The standard Atari DOS RUN and INIT vectors are NOT acted on.

LOCK [Dn:] and UNLOCK [Dn:] pg32
    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 disks. UNLOCK disables the LOCK command.(Only affects 2.x disks).

MDUMP [address [#bytes]] [/P] pg59
    External-1.x and 2.x This utility will display memory locations in HEX/ATASCII/ASCII format. It is very similar to DUMP but works on blocks of memory rather than files.

MEM pg53
    Internal-2.x Displays MEMLO and MEMHI.

MEMLO pg53
    External-1.x and 2.x Displays only MEMLO.

MENU [R][n] pg36
    External-2.x This command give you most of the features of the command processor but in a menu form. Its capable of single and multiple file functions.

MIOCFG [Dn:][path>]fname.ext [/SLN] pg102

       S - save
       L - load with Ramdisk format
      LN - load without Ramdisk format
    External-1.x and 2.x This utility command saves and reloads MIO configurations.

OFF LOAD [Dn:][path>]fname[.ext] offset [/SNPQ] or
OFF LOAD -R address [Dn:][path>]fname[.ext] pg60
    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. It may also be used to create non-relocatable versions of OFF_LOAD.

PAUSE pg48
    External-1.x and 2.x To temporarily halt execution of a batch file and

    to prompt the user for a response (key press) to continue.

PORT [path>]fname[.ext] pg46
    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 pg50
    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.

PROKEY pg104
    External-1.x and 2.x This utility command shell adds 20 programmable function keys.

PROTECT [Dn:][path>]fname[.ext] pg31
    Internal-2.x This command protects (locks) files from accidental erasure. This only affects 2.x disks.

PUTRUN [Dn:]fname[.ext] pg60
    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 and INIT vectors are used).

RDBASIC Dn: (XL/XE computer with internal BASIC on required) or
RD130 Dn: (130XE computer required) or
RDAXLON Dn: (Axlon RamPower 128 in Atari 800 required) pg20
    External-1.x and 2.x with restrictions These commands install a RamDisk device (electronic disk) in the place of a 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. See page 122 for SpartaDOS 3.2 Ramdisk files.

READ [Dn:][path>]fname.ext
    External-all Puts ANY file to screen, one screen at a time until done. Press space bar for next screen. Press 'Q' to quit any time.

RENAME [Dn:][path>]fname[.ext] fname[.ext] pg30
    Internal-1.x and 2.x Changes the name of an existing file or files.

RENDIR [Dn:][path>]oldname newname pg101
    External-1.x and 2.x This utility command renames subdirectories.

RPM [Dn:] pg55
    External-1.x and 2.x To display the drive speed in RPMs for user information.

RTIME8 pg122
    External-all Installs 'Z:' handler for use with R-Time 8 cartridge. For BASIC time access (pg130).

RS232 pg45
    External-1.x and 2.x   Loads the RS232 handler for the 850 interface.

RUN [address] pg57
    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 pg57
    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.

SCOPY Dn:[[path>]sourcefname] [/UR] Dn:[[path>]destfname] [/UR] pg124

       U - US Doubler sector skew copied
       R - Ramdisk identifier
    External-all   Compacts diskette into file, expands file into diskette
    and performs straight sector copies.

SET [mm/dd/yy] [hh/mm/ss] (for use with TIME command) or
TSET [mm/dd/yy] [hh/mm/ss] (for use with TD or XTD commands) pg41
    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.

SORTDIR [Dn:][path>] [/NTSDX] pg103

       N - name (fname)
       T - type (extension)
       S - size
       D - date
       X - reverse sorting order

        External-1.x and 2.x   This utility command sorts directories by
    name, extension, size and date.

SPCOPY or XCOPY pg27
    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) or
TIME[X] (for use with system clock) or
XTD (for use with R-Time 8) pg42 or
TIME (to display and set time for 3.2+ only) or
TD ON/OFF (turns on display line for 3.2+ or
TDLINE (installs external display line TD controls. 3.2+ only) pg121, 122
    External-1.x and 2.x, Internal-3.2+   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. 3.2+ - see pg 122.

TREE [Dn:][path] [/F] pg24
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] pg49
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]] pg30
External-1.x and 2.x To restore a file that has been erase.

UNPROTECT [Dn:][path>]fname[.ext] pg32
Internal-2.x This command unprotects (unlocks) files from accidental erasure. This only affects 2.x disks.

VDEL [Dn:][path>]fname.ext pg102
External-1.x and 2.x This utility command deletes many files at once with wild cards, query and report.

VERIFY ON or VERIFY OFF pg63
Internal-2.x This command changes the write mode to write with verify or write without verify.

WHEREIS [Dn:]fname.ext [/D] pg102

D - display
External-1.x and 2.x This utility command will find a full or partial file name anywhere on your drives. Option 'D' displays all matches in full SpartaDOS type directory listing.

XDIV pg51
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.

XKEY pg52
External-1.x and 2.x To install a 32 character keyboard buffer for XL/XE computers.

XINIT pg16

External-1.x and 2.x This is the command to initialize (format) a SpartaDOS 2.x disk.

ZHAND pg123
External-3.2+ ONLY Installs 'Z:' handler for use with R-Time 8 cartridge. For BASIC time access (pg130).

APPENDIX C___Table of all SpartaDOS Command Processor Commands

page	Command Name	Internal Cmd V1.x	Internal Cmd V2.x	Works w/2.0 Disk	External Sparta Command	May Use RUN to reEnter	Locates Itself at MEMLO	Remains Resident at MEMLO	Initial Load Address
22	?DIR	no	yes	no	no				
18	AINIT	no	yes		no				
58	APPEND	yes	yes	yes	no				
45	AT_RS232	no	no		yes	no	yes	yes	$5000
127	AUTOBAT	3.2+	only		yes	no	no	no	$5000
15	BASIC	no	yes		no				
20	BOOT	no	yes	no	no				
53	BUFS	yes	no		no				
126	BYPASS	no	no		yes	no	yes	yes	$3000
15	CAR	yes	yes		no				
54	CHKDSK	no	yes	yes	no				
43	CHTD	no	no	no	yes	no	no	no	$5000
31	CHVOL	no	no	no	yes	no	no	no	$5000
112	CLEANUP	no	no	no	yes	no	no	no	$2600
25	COPY	yes	yes	yes	no				
22	CREDIR	yes	yes	no	no				
23	CWD	yes	yes	no	no				
121	DATE	3.2+	only		no				
23	DELDIR	yes	yes	no	no				
14	DIR	yes	yes	yes	no				
14	DIRS	no	yes	yes	no				
51	DIS_BAT	no	no		yes				$5000
107	DISKRX	no	no	yes	yes				$3000
103	DOSMENU	no	no	yes	yes				$4000
58	DUMP	no	no	(1)	yes	no	no	no	$5000
28	DUPDSK	no	no	no	yes	yes	yes	no	$5000
29	ERASE	yes	yes	yes	no				
19	FORMAT	no	no		yes	yes	no	no	$6000
16	INIT	no	no		yes	yes	no	no	$6000
52	KEY	no	no		yes	no	yes	yes	$5000
123	KEY	3.2+	only		no				
56	LOAD	yes	yes	yes	no				

page	Command Name	Internal Cmd V1.x	Internal Cmd V2.x	Works w/2.0 Disk	External Sparta Command	May Use RUN to reEnter	Locates Itself at MEMLO	Remains Resident at MEMLO	Initial Load Address
32	LOCK	no	yes	no	no				
59	MDUMP	no	no		yes	no	no	no	$5000
53	MEM	no	yes		no				
53	MEMLO	no	no		yes	yes	no	no	$5000
36	MENU	no	no	yes	yes	no	yes	(2)	$5000
102	MIOCFG	no	no	yes	yes	no	no	no	$4000
60	OFF_LOAD	no	no	(3)	yes	no	yes	no	$B400
48	PAUSE	no	yes		yes	yes	no	no	$5000
46	PORT	no	no		yes	no	no	no	$5000
50	PRINT	yes	yes	yes	no				
104	PROKEY	no	no	yes	yes	no	yes	yes	$4000
31	PROTECT	no	yes	yes	no				
60	PUTRUN	no	no	yes	yes	no	no	no	$5000
20	RD130	no	no		yes	no	yes	yes	$3C00
20	RDAXLON	no	no		yes	no	yes	yes	$3C00

page	Command Name	Internal Cmd V1.x	Internal Cmd V2.x	Works w/2.0 Disk	External Sparta Command	May Use RUN to reEnter	Locates Itself at MEMLO	Remains Resident at MEMLO	Initial Load Address
20	RDBASIC	no	no		yes	no	yes	yes	$5000
83	READ	no	no	yes	yes	no	no	no	$3000
30	RENAME	yes	yes	yes	no				
101	RENDIR	no	no	no	yes	no	no	no	$5000
55	RPM	no	no		yes	no	no	no	$5000
122	RTIME8	no	no	no	yes	no	no	no	
45	RS232	no	no		yes	no	yes	yes	$5000
57	RUN	yes	yes		no				
57	SAVE	yes	yes	yes	no				
124	SCOPY	no	no	yes	yes	no	no	no	$3000
41	SET	no	no		yes	yes	no	no	$5000
103	SORTDIR	no	no	no	yes	no			$4000
27	SPCOPY	no	no	yes	yes	no	no	no	$3000
42	TD	no	no		yes	no	yes	yes	$5000
122	TD	no	no	no	no	no			
122	TDLINE	3.2+	only	no	yes	no	yes	yes	$5000
40	TIME	no	no		yes	no	yes	yes	$5000
121	TIME	3.2+	only	no	no	no			
24	TREE	no	no	no	yes	no	no	no	$5000
43	TSET	no	no		yes	yes	no	no	$5000
49	TYPE	yes	yes	yes	no				
30	UNERASE	no	no	no	yes	no	no	no	$5000
33	UNLOCK	no	yes	no	no				
32	UNPROTEC	no	yes	yes	no				
102	VDEL	no	no	yes	yes	no			$5000
63	VERIFY	no	yes		no				
102	WHEREIS	no	no	yes	yes	no			$3000
28	XCOPY	no	no	(4)	yes	yes	yes	no	$5000

51 XDIV	no	yes	no				
52 XKEY	no	no	yes	no	yes	yes	$5000
16 XINIT	no	no	yes	yes	no	no	$4180
42 XTD	no	no	yes	no	yes	yes	$5000
123 ZHAND	3.2+	only	yes	no	no	no	$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 and 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 the COMTAB location is stored at DOSVEC (locations 10 and 11).


```
COMTAB+13=location of day
COMTAB+14=location of month
COMTAB+15=year
COMTAB+16=hours (24 hour format)
COMTAB+17=minutes
COMTAB+18=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 and 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's menu	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 (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 configuration 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.

RS232 Handlers are included for communications using the Atari 850 interface or the ATR8000 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 1050 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 1050s 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 dealers 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 turn around is general 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/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 needle nose pliers
20-35 watt small tipped soldering iron (optional)

COVER REMOVAL
Turn the 1050 on its back and remove the 6 screws. (4 are recessed and two are on the front). Place the screws into your dish.

Carefully turn the drive back on 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 off as one piece. Set these aside.

THINGS TO LOOK FOR
Notice how the drive assembly sits in the case and note the 4 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 4 steel pins at the center of these washers which fall out during disassembly of the early 1050 drives (they are glued in on 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 will need to make a chart indicating the color pattern for each connector before you unplug them.

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 needle nose 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 4 rubber spacers and the 4 steel pins (if they're loose) in your parts dish.

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, 4 plastic tabs or two plastic tabs, or 3 small Phillips screws and 3 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. 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 4 jumpers (JP1-JP4) determines which type of U10 chip you will need. We're not 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/JP4 should be closed (jumpered). If the replacement U10 does NOT have a paper label, then JP2/JP4 should be open (no connection, JP1/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 handing 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 possession. Do not attempt this modification unless you feel confident with a soldering iron. The other jumpers JP5/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 are 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 its 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, and then lower the front of the PCB. The PCB should easily snap in place under the plastic tabs. (Install the 3 washers and screws if your drive had them).

REINSTALL THE RUBBER WASHERS and STEEL PINS (IF REMOVED)

Press the 4 rubber washers with the recessed side down, onto the plastic posts in the front half of the drives case. The 4 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 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
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 didn't unplug your drive from the PCB, you can skip this instruction.

REPLACE THE TOP COVER
To replace the top cover, first line up the bezel over the front of the drive frame, and 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 drive upside down and lay it on its back. Screw the 6 screws back into place and presto!

You're Done!

STARTUP and TESTING
Plug the drive back into your system. If you are 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 disks 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.

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 cant get it to work, send your drive 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.

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, and then back on (cold start) to re-initialize the internal format settings. Failure to do this could create format errors with the other DOS.

APPENDIX G___US DOUBLER INTERFACE

The following is a list of the SIO commands of the US Doubler and their usage.

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 Atari drive. The data frame (sector size) received will depend upon the density/size of the disk. The US Doubler automatically adjusts to a new sector size when a disk is inserted into the drive. Sectors 1-3 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 drive. The data frame (sector size) sent is dependent upon the density/size of the disk. The Doubler automatically adjusts to a new sector size when a disk is inserted. Sectors 1-3 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 though 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 4 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 controllers 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 the sector may not be written either).
Bit 4: RECORD NOT FOUND--The sector doesn't exist if this bit is 0 (low).
Bit 5: RECORD TYPE--If 0 (low), a special write command was given when last sector was written. A normal drive won't create this type of sector (unmodified). Note the data is correct, its a method of protection some publishers use.
Bit 6: WRITE PROTECT--If 0 (low), the disk was write protected. 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--Drive door is open if low (0).

Byte 1: The status the CPU generates indicating the following things:
Bit 0: COMMAND FRAME--A 1 (high) indicates 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 DOUBLER-always 0.
Bit 3: WRITE PROTECT--The disk is CURRENTLY write protected if this bit is a 1.
Bit 4: MOTOR ON--The disk is CURRENTLY spinning if this bit is a 1.
Bit 5: SIZE--The sector size is 256 bytes (double density) if this bit is a 1.
Bit 6: --not used--
Bit 7: 1050 Enh mode--This bit is 1 if in DD, but the sectors are 128 bytes long (enhanced density).

Byte 2: The timeout value used when formatting by the computers SIO

routine.

Byte 3: --unused-always zero--

FORMAT DISK (General Format Command)
Command: !($21)
AUX 1: not used
AUX 2: not used

Notes: This command formats a disk in either double or single density. A data frame of 128 bytes (if 128 byte sector format) or of 256 bytes (if 256 byte sector format) is returned. The DOUBLER doesn't return the bad sector list. The first two bytes in the data frame will be $FF $FF.

FORMAT DISK (1050 enhanced density)
Command: '($22)
AUX 1: --not used--
AUX 2: --not used--

Notes: This command formats a disk in 1050 enhanced density. A data frame of 128 bytes is returned. The DOUBLER doesn't return the bad sector list. The first 2 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 disk in single/double/enhanced density modes AND allows the user to specify the sector ordering. The computer sends a data frame of 128 bytes to the drive. The first 12 bytes are the configuration bytes, and the next 18-26 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.

Enhanced 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:
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

Enhanced 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 the 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. The computer sends the drive a 12 byte data frame which consists of the following:

+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
+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)
AUX1: --not used--
AUX2: --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___DISKS

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. The raw SSSD format yields 92160 bytes per disk (90 KB).

1050 'Enhanced Density' (1050ED) consists of 40 tracks of 28 sectors each. Each sector also has 128 bytes but is shorter in physical length. The raw 1050ED format yields 143360 bytes per disk (140KB).

Single Sided, Double Density (SSDD) uses 40 tracks of 18 sectors each but each sector stores 256 bytes. The raw SSDD format yields 184320 bytes per disk (180KB).

APPENDIX I___GLOSSARY

ADDRESS	a location in memory.
APPEND	to add on to.
ASCII	American Standard Code for Information Interchange which uses 7 bits to define a one byte code from 0- 127 decimal (0-$7F).
ATASCII	A superset version of ASCII used only with the Atari computer.
BANK	predetermined size block of memory.
BATCH	a batch file is a file containing a group of commands to be executed consecutively.
BAUD RATE	speed for data transmission.
BINARY	the BASE 2 numbering system.
BIT	a binary digit-either a 0 or 1

BOOTUP refers to system INIT which sets up the computer when powering up.

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 0-255 (0-$FF in 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

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 you are.

DATA information generally used or operated on by a program.

DEBUG to isolate and eliminate errors from a program.

DECIMAL Base 10 numbering system, not very useful to computers unless translated to Hex or binary, but easy for humans to understand.

DEFAULT standard condition or value that exists upon running a file.

DENSITY generally the number of bytes per sector is the disk density, single being 128, double being 256 bytes per sector. Actually density specifies the number of bytes per track on a disk.

DEVICE Atari devices are Dn:,E,S,R,P,C, referring to 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 list of all files.

DOS Disk Operating System, 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 disk.

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.

HANDLER a program written to handle a device.

HARDCOPY printed on paper.

HARDWARE computer, disk drives, printers are hardware..programs are 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 base 16 numbering system which includes 16 unique single digits used for counting.

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.

I/O Input/Output, this is what ties a computer to the outside world.

IOCB Input/Output Control Block. A 16 byte block of reserved memory which acts as a parameter passing window for I/O functions.

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 to program. BASIC, LOGO, PASCAL, Assembler are all languages.

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 to a form which is compatible with phone systems and vice-versa.

NESTED fitted within similar things.

OS Operating System, usually a ROM based, machine language program that handles interrupts for the screen, keyboard etc., maintaining interaction between all devices, allowing the computer to work in the first place.

PATH trail or course taken from one place to another.

PARALLEL the transfer, processing or manipulation of all the bits in a byte simultaneously by using separate lines for each bit.

PERIPHERAL external device connected to your computer.

PORT a place of access to a system. ie. the serial communications port or the parallel joystick ports.

PROGRAM a set of instructions to tell the computer how to accomplish some certain task.

PROMPT a signal to the user that some action may be needed by them.

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.
ROM	Read Only Memory. Permanent memory that can only be read.
RS-232	communications interface standard designated by the Electronics Industries Association (EIA)
SECTOR	block of storage used on disk, can be 128 or 256 bytes with Atari formats.
SERIAL	data transfer occurring on one signal line. The data bits are sent down the line sequentially.
SOFTWARE	programs 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, how densely data can be packed on a disk.
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 a 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 chars 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.

SPARTADOS TOOL KIT

This is an incredible collection of new, unreleased utilities written for all SpartaDOS versions. These are solid tools all written by the professional programmers at ICD. A few utilities may not be applicable to the older SpartaDOS versions. SpartaDOS ToolKit is a must for any serious SpartaDOS user. SpartaDOS ToolKit will help you get the most power out of SpartaDOS--the most powerful DOS for 8-bit Atarii!

The included tools are:

RENDIR	renames subdirectories pg101
VDEL	verify delete (prompts you to delete a file or not) pg102
WHEREIS	find a filename fast (full or partial) anywhere on your drives pg102
MIOCFG	save and reload MIO configurations. pg102
SORTDIR	sort directories by name, extension, size and date. pg103
DISKRX	the SpartaDOS disk editor - edit sectors, trace files or sector maps in any density, rebuild directories, files etc. Powerful! pg107
CLEANUP	detects SpartaDOS file structure defects, allows rebuilding of directory structure. pg112
DOSMENU	a SpartaDOS menu for Atari DOS 2 lovers (painful for Command Processor lovers) pg104
PROKEY	adds 20 'pf' (programmable function) keys, used for path prompt, screen color change, IBM style recall console keys and more to SpartaDOS. pg104

RENDIR Command

Purpose - This is a simple external command to rename SpartaDOS subdirectory names. Only the directory name is changed, there is no effect on the contents of the directory.

Syntax
RENDIR [Dn:][path>]oldname newname

Remarks
'Oldname' and 'newname' may include extensions as desired. RENDIR only allows valid SpartaDOS filename characters. No wildcards are allowed. RENDIR is not case sensitive, lowercase is converted to uppercase and inverse is converted to

non-inverse.

VDEL Command
Purpose - This is a simple command which prompts you whether or not the file should be deleted.

Syntax
VDEL [Dn:][path>]filename.ext

Remarks
The 'filename.ext' will usually include or be replaced with wildcards. That is when VDEL becomes most powerful. It will find any filename matches, display each one and prompt you for a 'Y' or 'N' to delete or not. <ESC> is also an option if you want to quit the procedure. At the end of the process, a message appears reporting the number of files deleted.

WHEREIS Command
Purpose - To quickly find a filename (full or partial) anywhere on your drives. This becomes especially useful on a hard disk with multiple partitions or any disk with subdirectories. WHEREIS can search all directories on all drives for filename matches.

Syntax
WHEREIS [Dn:]filename.ext [/D]

D - display

Remarks
The 'filename.ext' may include wildcards as desired. Any and all matches found will be displayed with the full path from the root directory to the filename match. The number of matches found will be displayed at the end of the search. Specifying the optional drive number 'Dn:' will limit the search to that specific drive, otherwise, all drives will be checked.
The optional 'D' parameter will display the matching filenames as in the SpartaDOS long form directory listing so you can see length, date and time, for each file.

MIOCFG Command
Purpose - To save and reload Multi I/O configurations. Multi I/O configurations may be saved as files and then reloaded as desired. This is especially useful for Multi I/O owners without hard drives since they previously had no means to save a configuration. Alternate configurations may

also be loaded as a quick alternative to manually changing the menu screens.

Syntax
MIOCFG [Dn:][path>]filename.ext [/SLN]

S - save
L - load with Ramdisk format
LM - load without Ramdisk format

Remarks
The 'filename.ext' may include wildcards as desired. The first match will be used. The 'S', 'L' or 'LN' parameter is required and selects the mode of operation. Like most other SpartaDOS commands with '/' parameters, a space is required between the end of the filename and the '/' character. COPY with '/A' APPEND is the only SpartaDOS exception. All Multi I/O configuration menu selections are saved in a file with the 'S' parameter.

When the file is reloaded using the 'L' mode, all MENU defaults are set up according to the file loaded. The Multi I/O Ramdisks are then automatically reformatted using the default SpartaDOS type double density format.

When the file is reloaded using the 'LN' mode, no format is executed. This mode should not be used if the Ramdisk starting and ending sectors will be changed.

SORTDIR Command
Purpose - To sort directories by name, extension, date or size. Directories can now be sorted quickly and safely! SORTDIR will quickly read the directory specified, sort it according to the mode selected and rewrite the directory in sorted order. Forward and reverse sorts are supported as well as double priorities in all modes (ie. TIME is a second priority to DATE when the '/D' mode is selected).

Syntax
SORTDIR [Dn:][path>] [/NTSDX]

N - name (filename)
T - type (extension)
S - size
D - date
X - reverse sorting order

Remarks
SORTDIR may be entered with the optional drive specifier or one of the optional mode parameters. If no drive specifier or mode parameter is entered, the list

of parameters and meanings is displayed. A second parameter 'X' may be included for a reverse (descending) sort. 'Sort Completed' is printed when the sort is finished.

When sorting by name, type (.ext) is the second priority. When sorting by type, name is the second priority. When sorting by size, filename is second priority with type as third priority. Numbers come before letters in sort order. Specifying the path will allow you to sort any directory on any drive from any other directory. Path, along with a simple batch file, will allow you to sort multiple directories easily.

DOSMENU Command
Purpose - A SpartaDOS menu for Atari DOS 2 lovers

Syntax
DOSMENU

Remarks
This is a painless way for Atari DOS 2.0 or 2.5 addicts to use SpartaDOS. DOSMENU is a Command Menu for SpartaDOS which loads in and replaces the Command Processor user interface. DOSMENU does take up some user RAM so some programs may not be compatible with it (enter 'Y' then MEM to find out where MEMLO is). Wildcards are supported by most commands. Either a comma or a space may be used as the delimiter when using Copy Rename, etc. A sample menu display is shown below.

```
     DOSMENU Ver 1.3 6-28-88
     (C) 1994 by FTe

     Directory 1-8 (Long), !-@ (Short)

     A. Disk Directory   L. Binary Load
     B. Run Cartridge    M. Run At Address
     C. Copy File        O. Make Directory
     D. Delete File      P. Pick Directory
     E. Rename File      Q. Kill Directory
     F. Lock File        T. Printer Off
     G. Unlock File      V. View File
     I. Format (Sparta)  X. Disk Info
     J. Duplicate Disk   Y. Do DOS Command
     K. Binary Save      Z. Reboot System


     Enter Command or Return For Menu:
```

The 'Format' command loads the SpartaDOS XINIT.COM file, so it must be available on a disk if you are going to format.

The 'Duplicate Disk' selection requires DUPDSK.COM and DUPDSK requires disks to be preformatted in matching densities (source and destination).

Most of the commands are using SpartaDOS XIO functions of internal SpartaDOS commands.

PROKEY Command
Purpose - This command shell adds 20 'pf' (programmable function) keys, used for path prompt, screen color change, IBM style recall console keys and more to SpartaDOS.

Syntax
PROKEY

Remarks
PROKEY.COM loads in and supports the SpartaDOS Command Processor with the

following EXTRA commands and features.

PF Keys The programmable function keys are used by holding down the control key and the appropriate number key <CONTROL + number>. A second bank of functions are selected by the addition of the SHIFT key <CONTROL + SHIFT + number>. These keys are programmed by typing 'PFn string' (where 'n' is the number and 'string' is the string of characters to be stored in the function key). Valid 'PF' numbers are 1-20. There may be up to 20 characters in the string. Use the '@' character at the end of a string to execute a <RETURN>.

CLPF Clears all 'PF' keys

PROKEY.BAT Batch files are a natural way to load the PF keys. Upon initialization, PROKEY looks for a file called PROKEY.BAT. You can keep alternate sets of keys stored for use with ACTION!, BASIC etc. See our example batch files.

<CONTROL S> If PF1 is loaded, <CONTROL + S> replaces its toggle function to start and stop screen scrolling.

<CONTROL C> If PF3 is loaded, <CONTROL + C> replaces its normal function (marking end of files from E:)

BELL The BELL command has been added to replace the normal <CONTROL + 2> function. You would use this in batch files as a warning indicator, etc.

Screen Color Entering either 'BLACK', 'GREEN' or 'BLUE', will change your display color which may help make it more readable (helps on monochrome also). We prefer 'BLACK' for good resolution on our inexpensive monochrome monitors.

COLD or EXIT The commands 'COLD' and 'EXIT' just do a 'cold start' of your computer system. This is the same as typing in 'RUN E477' which many of you already know. A cold start is about the same as turning the power off and on to your computer except for two distinct differences.

- o There is no waiting required on an expanded memory XL/XE for the RAM chips to bleed down and lose their memory.

- o The internal Ramdisk data is still there. You can get at it by using RD.COM with the 'no format' parameter.

HELP or ? The commands 'HELP' and '?' give you a brief help menu with a list of the available commands and/or features.

PATH This is similar to the 'PROMPT' command in MSDOS. 'PATH ON' or 'PATH OFF' are the two valid commands. With 'PATH ON', the directory path is displayed as part of the 'Dn:' prompt. After every <RETURN>, a '?DIR' type query is done through SpartaDOS, the drive is read and the path displayed.

IBM Mode If IBM mode is on ('IBM ON'), PROKEY will emulate the use of cursor keys like MSDOS does. Each key press operates on the 'last line buffer', that is, the last command line that you entered into PROKEY. To turn IBM mode off, type 'IBM OFF'.

These special editing keys are as follows:

<Right arrow>	The right arrow key will pull the next character from the last line buffer and place it into the current line.
<Left arrow>	Will backspace one position (identical to the <BACKSPACE> key).
<CONTROL + INSERT>	Will place you into 'insert mode'. All key presses will be processed without advancing the last line

	buffer's index.
<CONTROL + DELETE>	Will advance the last line buffer's index thereby 'deleting' characters from the last line buffer (NOT the current line!).
<START>	Will repeat the remaining characters in the last line buffer. If you are in the first position of the input line, pressing <START> will repeat the entire last line.
<SELECT>	Works like the <START> key, except that the next word only is pulled from the last line buffer.
<SHIFT + DELETE>	The <SHIFT + DELETE> key will erase the entire current input line, placing the cursor back in the first position of the line.
CLS	The command 'CLS' takes the place of the <SHIFT + CLR> (clear screen) function which is lost with 'IBM ON' command mode

EXAMPLE OF IBM MODE

Let's say that the last command line executed looked like this:

COPY D1:DOS>PROKEY.COM D3:PROKEY.*

Now, let's assume that you also need to copy the PROKEY.DOC file, too. Instead of keying in the entire line again, just do this:

- hit the right arrow key 21 times
- key in DOC
- press the <START> key

DISKRX Command
Purpose - To edit sectors, trace files or sector maps in any density, rebuild directories, files etc.

DiskRx is the ICD SpartaDOS sector editor. Most of its functions are for SpartaDOS disks, however, it can also be used as a basic read/write editor for non-SpartaDOS disks.

DiskRx operates in two modes: Disk mode and File mode. File mode may be used

on SpartaDOS disks only. In Disk mode, the sectors of any disk are accessible sequentially or by random access (sector number input). In File mode, only those data sectors belonging to the specified file may be viewed.

Syntax
DISKRX or
DISKRX Dn: or
DISKRX [Dn:][path>]filename.ext

With the first option, DiskRx will prompt you to select a drive.
With the second option, the drive number is pulled from the command line.
Both of these options start DiskRx in the Disk mode.
The third option begins DiskRx in File mode as pulled from the command line, if found. Otherwise, it begins in Disk mode with the drive that was specified in the filename. If 'Dn:' was not specified it is assumed to be the default drive (the drive number in the DOS prompt). It is not necessary to specify a path name, DiskRx will find the first occurrence of a file unless a path is given.

Remarks
Once DiskRx is finished loading, the Main Screen is displayed. At the bottom of the screen is a line showing how to get to the Menu or exit the program. Above that is the prompt line where various messages are displayed. The information area seen above the prompt line shows the current mode, drive number or file name, sector number, sector type, bytes per sector, whether or not the disk is a SpartaDOS disk and whether or not the current sector is allocated in the SpartaDOS bitmap. If in File mode, the first sector map, first data sector and index into the file of the current sector are also shown.

The sector type is represented by a four-letter abbreviation. Seven types are recognized:

- BOOT - sectors 1 through 3
- BMAP - bitmap sector
- DIRM - sector map of main directory
- MDIR - main directory
- SUBM - sector map of subdirectory
- SDIR - subdirectory
- DATA - all other sectors

Only BOOT and DATA are shown on non-SpartaDOS disks, or disks on which the directory tree cannot be mapped.

Bytes per sector (b/s) is 128 or 256. SpartaDOS disks show a 'Y' in the 'Sparta' field. The bitmap allocation of a sector on a SpartaDOS disk is shown in 'alloc' by a 'Y' or 'N'.

Above the information area is the data area. Two fields are displayed, the bytes in the current sector are shown on the left in their Hex form and on the right as ATASCII characters. At the far left side of the Hex field the byte index is shown. This ranges from '00:' to 'F8:' for 256 byte sectors, or from '00:' to '78:' for 128 byte sectors. Eight bytes are shown in each row across the display. 128 byte sectors are displayed entirely on one screen and 256 byte sectors occupy two separate screens. Press '>' to view the second half of a 256 byte sector, '<' returns to the first half.

Two Menu screens are available which show the available commands. Once familiar with the program, you should not need these.

You will find DiskRx easy to use. In most cases, prompts will guide you to the next step. The following commands are valid from the prompt cursor:

(A)Arithmetic Conversion - Converts decimal, hex and binary numbers. Default is decimal, to indicate binary - enter 'B', to indicate hex - enter 'H' or '$'. Press <RETURN> or <SPACEBAR> after entering the first number, then enter 'D'(decimal), 'B'(binary) or 'H or $'(hex) to indicate the desired conversion. Trivial conversions, ie. binary to binary, are ignored.

(B)Blank sector - Blank the sector in the buffer which may then be edited and written if so desired.

(C)Change disk - Change disk, drive or mode. When editing a SpartaDOS disk, 'C' forces a remapping of the directory tree, when rebuilding a damaged directory, use this command to see how well you are doing. If in File mode, 'C' forces a change to Disk mode. You may enter '1' through '8' to select a drive, or indicate the default drive by a <SPACEBAR> or <RETURN>. The default is the drive shown in the DOS prompt.

(D)Directory - Valid only on SpartaDOS disks, it will display all directory entries on a disk whether or not they are presently valid or in use. Optionally, a subdirectory path may be selected. Note that in DiskRx, all path names begin at the Main directory. The directory display shows:

- the full name of the file and its size
- the present status of the entry
- the first sector map of the file
- the sector(s) where the entry is actually written

The status codes are displayed on the screen bottom. If a path was specified, it is shown near the top of the page. Some of the status options shown are not valid under SpartaDOS 3.2 or earlier, but will be in effect with the SpartaDOS X cartridge, these are Archive, Hidden and Open.

(EA)Edit ATASCII and
(EH)Edit Hex - The cursor is positioned in the ATASCII or Hex fields of the current sector. The arrow keys are used to move the cursor to the desired position, editing is terminated by the <ESC> key. If editing the Hex field, only '0' through 'F' may be entered, in ATASCII edit, all characters may be entered except the cursor movement and <ESC> characters. To enter these characters, move to the Hex field and enter the Hex equivalents. Note that the sector display is instantly updated, and all characters changed are highlighted in the Hex field. To terminate Edit, press <ESC>. Note that changes are not permanent until the sector is written with a 'W' command.

(F)File mode - Either specify a file name (optionally with a subdirectory path) or leave file mode by pressing <ESC> or <RETURN>. You can also use the 'F' key to change drive numbers by entering only 'Dn:'

A path is assumed to begin at the main directory. Paths are not shown on the information field as there is not sufficient room. Note that the program will find a file without a path name, you need to specify the path only in the case of multiple copies of the same file on the current disk. You can list all occurrences of a file with the directory command. Due to space limitations in the prompt line, only 38 characters may be used to show a path/filename. Use '*' abbreviations to access longer path names.

If a 'Dn:' is not specified in the file name, the currently selected drive is used. You may also view the sectors of any directory by typing the directory name. The Main directory is referenced by the name 'MAIN'. A file may only be rewritten on sectors already allocated.

(H)Hex - This command toggles the numbers in the information and prompt fields between Hex and decimal display.

(M)Menu - In Menu, change page with 'P' and exit with <ESC>. All commands may be run from the menu and all but 'H', 'P' and <CONTROL + P> will get you out of the Menu to the appropriate screen and command.

(O)Override parameters - Useful in disk recovery where the first sector has been damaged and the basic status of the disks is not readable or is incorrect. Use some caution as you may get into trouble by injudicious use of override. You can change:

- SpartaDOS status
- maximum disk sectors
- bytes per sector
- disk write locked parameters

Optionally, these parameters may be rewritten to sector 1, assuming that it is not a bad sector.

(QP) Quit Program - Quit DiskRx and re-enter SpartaDOS.

(R)Read sector - Read a sector from diskette. You have three options:

- o R Read next sector in sequence
- o Rn Read sector number 'n'
- o RL Read Last sector

In Disk mode, the 'raw' sector number is specified. In File mode, the sector number relative to the beginning of the file is specified.

(SH)Search Hex and
(SA)Search ATASCII - Search from the current sector through last sector of either the disk or file (depending on current mode) for a sequence of bytes. Wildcards or unknown characters may be specified by a '?' in the character sequence entered. The search string may be up to 18 bytes in length. If a match is found, the sector is shown and the index into the sector where the string begins is printed in the prompt field. You are asked if you want to continue the search. The <ESC> key will terminate the search at any time.

(T)Toggle allocation - Allocates or deallocates current sector in the SpartaDOS bitmap.

(W)Write sector - Write displayed sector to disk. You have two options:

- o W Write data to the current sector
- o Wn Write data to sector number 'n'

In Disk mode, the 'raw' sector number is specified. In File mode, the sector number relative to the beginning of the file is specified.

(>)Read next and
(<)Read previous - Read next or previous sector of file or disk. If 256 byte sectors, read either the alternate half of the sector or the first half of the next (or last half of the previous) sector. If 128 byte sectors or in sectors 1 through 3, read the next (or previous) sector.

When examining a sparse file (one in which not all sectors have been actually allocated), any such sectors will show a blank sector in the data field, and 'Sector not allocated' will be displayed. No file index, bitmap allocation or sector number will be shown in this case.

(+)Scan forward and
(-)Scan backward - Quick Scan forward (or backward), until end (or beginning) of disk or file, or until a key is pressed. In File mode, only allocated sectors are displayed.

(1)Recover - Write tagged sectors to a file on another disk. Useful in file

recovery. After locating the desired data, tag the sector with <SPACEBAR> to write it to the new file specified. A sector may not be written more than once. Terminate the function and close the file with a '1' or by leaving the program. Note that the new file is written only to a different drive, because the main use of this function is in recovering text or data from a disk with a severely

damaged directory.

(2)Recover - Point to a sector map and write a file to another disk from the map. Again, this is useful when a directory has been destroyed.

(3)Recover - Create a new directory entry from a sector map. The name and optional path are entered. The new entry is always written at the end of the directory chosen, remember that a SpartaDOS directory may have a maximum of 127 entries (this limitation does not apply to SpartaDOS X). The new entry is given a status of 'in use' and you must modify this if wanting to protect it, create a subdirectory, etc. The current system time and date is used. The end of the directory is denoted by a status byte of zero. This function writes at that point, so if you want a directory entry at a certain place, use the edit function to put a zero in the correct position. See the SpartaDOS manual Chapter 19 for more detail.

(4)Recover - Write the current contents of the main buffer to any sector of any disk. This is most useful if you have a bad sector one on a disk. You can get a good copy from another SpartaDOS disk and use this command to write it to the faulty disk. Then you can edit the disk parameters in your new sector one as required for the target disk. Another possible use is capturing system sectors in a file for study or modification. See the SpartaDOS manual Chapter 19 for more detail.

<CONTROL + P> - Toggles the printer on and off. Only sector reads and directories are printed. If a printer is not on line, this menu option won't be displayed.

<ESC> - <ESC> is the general purpose get out and terminate command. You leave the Menu, Override and Directory screens, and the Edit function with this key. It will also get you out of any place you don't want to be (within reason).

Other DiskRx Notes

Two types of error messages are used in DiskRx. The message 'Disk error# (num) at sector (num)' refers to errors on the disk being edited. Usually this will be either and error 138, for a drive not on line or door open, or an error 144 for a bad sector. The message 'System Error# (num)' refers to an error detected from SpartaDOS and would almost always occur either when using File

Recovery functions 1 and 2 to write to another disk, or when specifying a file name with a directory path.

When editing a non-SpartaDOS disk, only Disk mode is valid. The 'D', 'F', 'T', '2' and '3' commands may not be used.

Upon entering the program, mapping of the disk directory tree is attempted. If the program appears to be locked or running wild, press <ESC> to exit. The directory will not be mapped and only BOOT and DATA sectors will be displayed. The usual reason for this condition is a messed up bitmap or directory sector map.

Similarly, upon entering File mode, mapping of the file is attempted. Again,

<ESC> terminates this if it appears unsuccessful. The main reason is generally a scrambled file sector map.

If the boot sectors are damaged, the program may not be able to determine the disk status. The message 'Unable to read boot sector' will be shown. You can then press <ESC> at the next drive prompt and use Override (O) to gain access to the disk.

The edit functions are used to repair damage. The file recovery functions can be used to retrieve 'lost' data or rebuild a directory from scratch by locating all the sector maps in a manual search. You can also use the program to 'patch' any file, customize prompts etc.

Study the SpartaDOS manual carefully for a complete explanation of the structure of a SpartaDOS disk, and use DiskRx to explore a good disk NOW, don't wait until you have a damaged system to learn DiskRx. This knowledge will enable you to rebuild or recover your trashed disks!

CLEANUP Command
Purpose - This program detects file structure defects on a SpartaDOS disk and alerts the user to their existence. In some cases, the user is able to correct the defects with CLEANUP. In other cases DiskRx, the ICD Sector Editor, will be required.

Syntax
CLEANUP Dn: [/P]

P - echo CLEANUP's output to printer

Remarks
'Dn:' represents the drive you want to inspect and/or correct. Optionally you may specify a '/P' from the command line to get the program's output echoed to

the printer.

At any time when you are asked by CLEANUP if it is OK to do something, respond with 'Y' key for 'yes' and any other key for 'no'.

CLEANUP first attempts to map the entire directory structure on the disk. During this process, it reports back on several types of findings:

(1) files or directories with invalid file name characters. This is a non-fatal, but often annoying, defect, as you will not be able to access these files.

(2) files with a first sector map of $0000. This is a non-fatal defect.

(3) files with a non-zero, non-valid sector map chain. This is a fatal type of defect in that if such a file is erased from DOS, a number of other files will be corrupted. The data will still be present, but some of the sectors will have been deallocated. If a disk write is subsequently performed, the good data will have been overwritten.

(4) files which, under SpartaDOS X, were opened for write or update and never properly closed. This is a fatal type of defect because the data in these files is not valid.

(5) directories with an incorrect length. This is a non-fatal error, but should be corrected, as SpartaDOS does use this information to test for the end of the directory.

If any flaws of types 2 through 4 are detected, you will be given the opportunity to mark the file as deleted. This means that the directory entry is marked 'erased' although the sectors are not deallocated as when erasing in DOS.

If a flaw of type 5 is detected, you will be given the opportunity to have CLEANUP correct it.

At the end of the mapping process, the total number of valid files and directories is reported.

Once the directory has been mapped, CLEANUP constructs a new bitmap of the drive. In a bitmap, each bit corresponds to one sector on the disk. If a bit is set (1) the sector is a free sector (not allocated). If it is clear (0) the sector is allocated to a file, directory, bitmap or boot sector. It is also no longer free.

During the process of building a bitmap, if any sector is claimed by more than one file or directory a collision has occurred. In this case, all the files

claiming the sector are shown and you will be given the opportunity to mark one as deleted. If one is so marked, the mapping process starts over. If you choose not to mark one deleted, the program regards this as a fatal type of error, because erasing one of the files may erase valid data actually belonging to the other file(s).

In some cases you may be able to decide which file or directory to keep, by its name or time and date stamp. Certainly, in many cases, it will be necessary to merely make a note of the names and use DiskRx to inspect each file before deciding which one to keep.

The mapping process and collision deletion continues until all collisions have either been resolved or ignored.

CLEANUP then compares the new bitmap with the drive's present bitmap. If any differences are found, you are asked if it is OK to print out a report. If your answer is 'Y', a report of the differences is displayed in the following format:

```
Bmap  Byte   Bit(s)        Disk Sctr
-------------------------------------
$0004 $01    1,3,4           $0008
                             $000A
                             $000B
```

Explanation

The difference was detected in bitmap sector 4, which is usually the first sector of the bitmap. The difference was found in the second byte of the sector (the first byte is $00). Bits 1, 3 and 4 differed from the original bitmap (bit 1 is the high order bit). The corresponding sectors on the disk are 8, 10 and 11.

CLEANUP then compares the new number of free sectors with the disk's original number. If there is a difference, it is reported.

If any fatal errors found have been resolved, you are given the opportunity to write both the bitmap and the number of free sectors permanently on the disk.

Finally, a series of cautionary messages may appear depending on the sort of errors found and (in the case of fatal errors) left unresolved.

CLEANUP will be most useful on hard disks with a large number of files - which sometimes become corrupted in one or more of the above ways. Such corruption is often due to such things as power failure or surges while using the disk, or pressing <RESET> while in the midst of a disk write.

Another possible cause is an untested or faulty program running wild! Programmers, note that it is usually recommended that you test new programs on a Ramdisk or a floppy until you are satisfied with their performance. Having your latest masterpiece trash your hard drive is guaranteed to start the day off wrong.

One last caution - there are certain 'buggy' public domain programs, 'pirated' programs and programs infected with a computer 'virus' available which are guaranteed to send your SpartaDOS disk to never-never land! When using an untested program for the first time, it is advisable to disconnect your hard drive. If you forget, it will probably be possible to use DiskRx and CLEANUP to reconstruct most of the disk - at the expense of several hours of painstaking work.

R-Time 8 SUPPLEMENT

The R-Time 8 now adds another dimension to the SpartaDOS Construction Set family which began with SpartaDOS version 1.1 and the US Doubler. Although the TD, XTD and TSET commands have been around for quite some time, they have been modified several times as problems arose. Their method of time and date support simply did not allow the flexibility that is now needed. As part of our ongoing development of SpartaDOS, we have developed a new SpartaDOS version which has built in TD and TSET functions. Accompanying the R-Time 8 is this new SpartaDOS version 3.2

The R-Time 8 package includes: 1) this supplement 2) the R-Time 8 cartridge 3) a double sided diskette with SpartaDOS 3.2 and updated command files on the front side. The back side is in Atari DOS 2 format and has the R-Time 8 generic 'Z:' handler (RTIME8.COM) and its source code (RTIME8.SRC).

The SpartaDOS Construction Set package includes: 1) a diskette containing the standard SpartaDOS version 1.1 on the front and public domain games with the LOGOMENU program on the back side 2) a diskette containing both SpartaDOS version 2.3 and 3.2 with their supporting command files 3) this supplement 4) the SpartaDOS Construction Set Owner's Manual. If you purchased the US Doubler, you receive the two US Doubler chips along with the SpartaDOS Construction Set package.

This manual is supplied with both the latest SpartaDOS Construction Set and the R-Time 8 package. The divisions of the manual and their contents are as follows:

Chapter 1___Introduction to the R-Time 8

This chapter is a brief introduction to the R-Time 8 cartridge. Explanations are included on how to use the R-Time 8 files with six different versions of DOS and which versions of DOS you may use with your computer.

Installation (Plugging it in)

Install the R-Time 8 cartridge as you would install any other cartridge for the Atari 8 bit computer line. The flap on top with the ICD logo faces toward the front of the 800 and 800XL computers. On the Atari 800 computers (which have both left and right slots), you may use whichever slot is most convenient (normally the right since most other cartridges use the left slot). Notice that the top of the R-Time 8 cartridge has an expansion port. You may use it to plug in your language cartridge (Pilot, ACTION, BASIC XE etc.) on top of the R-Time 8.

Booting Your Computer

First you must choose which DOS to use. Although any DOS should work, your choice governs the flexibility and ease of use of the R-Time 8. For example, if you use Atari DOS 2.5, you lose the powerful TIME, DATE and TD commands supported from SpartaDOS 3.2 Most importantly, NO VERSION OR DERIVATIVE OF ATARI DOS 2 SUPPORTS TIME AND DATE STAMPING OF FILES. This was the whole premise for making the R-Time 8 - to time and date stamp SpartaDOS files. A standard 'Z:' handler is implemented for which ever DOS you choose. With this interface to the cartridge, you may use BASIC (or whatever language you wish to use) to access the R-Time 8 for a wide variety of applications.

DOS Choices and Their Use with the R-Time 8

SpartaDOS 3.2 (for use with 800XL, 1200XL, 130XE and most Operating Systems
If you have one of these computers, this is the choice to make. The Command

Processor has the commands TIME, DATE and TD built in, which allow you to read and set the time and date. TD allows you to turn on and off a time and date display line, however, the actual handler code (TDLINE) must be loaded before the first use of the internal TD command. To install the 'Z:' handler, you must enter the command ZHAND.
NOTE: ZHAND is only necessary if you wish to address the R-Time 8 as the 'Z:'

device. This is generally for easy access from BASIC or other language programs.

SpartaDOS 2.3 (for use with 800XL or 130XE using the standard OS only)
This version of SpartaDOS has a MEMLO of under $0E00, which gives almost an extra 4000 bytes of free user memory Atari DOS. Use the external TD or XTD commands to provide the interface between SpartaDOS and the R-Time 8. The external TD command also provides the time and date display line. Neither of these supplies the 'Z:' handler, but they take less memory. An alternative to TD or XTD is the RTIME8 command. This supplies the 'Z:' handler as well as the DOS to R-Time 8 interface. The TSET command is used to set the time or date and may be used while either RTIME8 or TD/XTD is installed.

SpartaDOS 1.1 (works with any 8-bit Atari computer with at least 32K RAM)
If you have an 800 or need a translator, use this version, if not - it is better to use version 3.2 or 2.3 The same commands work for this version as for SpartaDOS 2.3 This version of DOS has a much higher MEMLO than 2.3 or 3.2 and is not capable of reading Atari DOS 2 diskettes directly. There are several different 1.1 versions (ie. NOCP, NOWRITE, SPEED and STANDARD) which work around its deficiencies.

Use one of the following DOS types if you don't have SpartaDOS
Atari DOS 2 or 2.5 (and its many followers ie. MYDOS, TOPDOS, SMARTDOS etc.)
You must rename the RTIME8.COM file to AUTORUN.SYS and boot your DOS 2 diskette. The back side of the diskette supplied with the R-Time 8 cartridge is an Atari DOS 2 format with the RTIME8.COM handler and the RTIME8.SRC source file. The handler will load and relocate itself at MEMLO.

NOTE: the time and date display line will automatically be turned off when you enter DOS (by the DOS command from BASIC).

DOS XL (by OSS)
To install the R-Time 8 handler, enter the command RTIME8 from the Command Processor. Since this is a CP (Command Processor) type of DOS, the time and date display line will remain on (as in SpartaDOS) when you enter DOS (by the DOS command from BASIC).

Atari DOS 3
If you have this DOS, we strongly recommend that you do not use it anymore. Go out and purchase the SpartaDOS Construction Set. Atari DOS 3, a product of the old Atari, will become a nightmare since: 1) it is not supported by Atari or any software company 2) it is very difficult to convert DOS 3 files to any other Disk Operating System (DOS).

Some Examples Using the 'Z:' Handler with BASIC

These examples all assume that you have BASIC installed in your computer (either internal or a cartridge) and wish to access the R-Time 8 in a program. For more details on the 'Z:' handler functions, see Chapter 5 of this supplement. To install the 'Z:' handler, choose a DOS and perform one of the following operations depending on your selection:

SpartaDOS 3.2

Boot the DOS and enter the commands:

```
TDLINE
ZHAND
CAR
```

SpartaDOS2.3, 1.1, DOS XL

Boot the DOS and enter the commands:

```
RTIME8
CAR
```

Atari DOS 2 (and family)

Move RTIM8.COM to your DOS diskette and rename it to AUTORUN.SYS

Now boot that diskette, the handler will install automatically

Turning the Clock On and Off

To turn the clock display on, enter the following command (from BASIC).

```
XIO 38,#1,0,0"Z:"
```

An extra display line will appear on top of the screen containing the correct time and date. Most programs from BASIC should work with the display line turned on, however, there are many exceptions. Practically all programs that use custom display lists or vertical blank routines will not work correctly. In most cases, the display line will simply be lost.

One inherent problem with Atari DOS 2 is the non-resident nature of the DUP.SYS program. The time and date display can't be on while using DUP (because DUP loads on top of the interrupt handler). Since it is inconvenient to always turn the display off before typing 'DOS', it is automatically done for you. There is absolutely no problem when using SpartaDOS or OSS DOS XL. The time and date will remain on since no DUP.SYS is needed.

To remove the time and date line from the display, enter the command:

```
XIO 39,#1,0,0,"Z:"
```

Reading the Clock Using BASIC

Your BASIC programs (or any language) may read the time and date in either formatted or unformatted form. This is accomplished by performing an XIO call followed by several GETs or an INPUT on an open IOCB. For example, to read the FORMATTED time, type and run the program:

```
10 DIM TIME$(10)
```

```
20 OPEN #1,4,0,"Z:"
30 XIO 32,#1,0,0,"Z:"
40 INPUT #1,TIME$
50 PRINT "Current Time is ";TIME$
60 CLOSE #1
```

Setting the Clock

Your BASIC program may set the time or date by performing an XIO and a sequence of three PUT statement. The following is an example of setting the date:

```
10 OPEN #1,8,0,"Z:"
20 PRINT "Enter Date (MM,DD,YY)";
30 INPUT MONTH,DAY,YEAR
40 XIO 35,#1,0,0,"Z:"
50 PUT #1,DAY : PUT #1,MONTH : PUT #1,YEAR
60 CLOSE #1
```

When setting the time, the hour must be given using a 24 hour clock where 12:00 midnight is 0, 12:00 noon is 12, 4:00 PM is 16 etc. The order of the PUTs is important. For date it is: day, month, year. For time it is: hour, minute, second.

R-Time 8 use from SpartaDOS

To use R-Time 8 with SpartaDOS version 3.2, study the commands in Chapter 3 of this supplement. To use it with version 1.1 and 2.3, review Chapter 12 and Appendix D in the SpartaDOS Construction Set Owner's Manual. For R-Time 8 support of Bulletin Board Construction Set 1.6 or earlier, use SpartaDOS version 2.3 and the XTD handler.

R-Time 8 Access from Machine Language

Although we strongly recommend that programmers using a standard DOS should access the R-Time 8 through the 'Z:' handler, we have included the source code for RTIME8 (RTIME8.SRC) on the back side of the R-Time 8 distribution diskette (Atari format). This gives detailed information on the inner workings of the clock chip to anyone familiar with machine language programming. Also refer to Chapter 4 of this supplement for information on the TIME and DATE vectors of SpartaDOS 3.2

R-Time 8 Maintenance and Service

This accurate timing device has been built to supply you with years of trouble free service. The accuracy is determined by the frequency of the crystal at location Y1 and the adjustor at YC1. To open the case, carefully pry it apart equally at both ends with a flat blade screwdriver. To adjust for greater accuracy turn the small slot at the center of YC1 slightly to the right or left. If adjustment is needed, calibrate the R-Time 8 once a week with an accurate watch and adjust accordingly if it is gaining or losing seconds. It should be possible to get very close with this method and a little patience.

Battery life is calculated for 3 to 5 years. The battery is a 200mah 3 volt Lithium cell. Replacement batteries are currently available for $5.00 including shipping. Full service including repair, calibration and battery

replacement is currently $20. This does not include damage due to abuse.

Chapter 2___Overview of SpartaDOS 3.2

This chapter briefly describes the changes between SpartaDOS 3.2 and SpartaDOS 2.3 Improvements have been made in the following areas:

- Better time and date support (internal TD, TIME, DATE commands)
- Internal R-Time 8 interface
- Internal JIFFY clock interface (for non-R-Time 8 users)
- Internal 32 character keyboard buffer (and KEY command)
- Automatic mini-buffer system for fast byte PUT and GET functions
- New vectors added for machine language support
- Control returned to DOS if DOS was active during RESET
- Supports both a STARTUP.BAT and an AUTORUN.SYS file
- Compatible with BASIC XE, 1200XLs and many modified Operating Systems
- BASIC ON/OFF command operation from within a batch file (not end only)
- NOISY I/O flag recognized
- Support for the Supra Hard Disk Interface
- All command entry in upper or lower case
- Full read capability for Atari DOS 2.5 type enhanced density format

Internal Real Time Support

Three new internal commands, time and date support vectors, an internal clock and access to the R-Time 8 cartridge, have been added for convenience and easy access.

Operating on the jiffy counter (at location $12 thru $14), the internal clock determines the number of jiffies (1/60th second) between the current and the last access. Time is then added to the internal clock based on this difference. Unfortunately a RESET zeros the jiffy counter, thus the time between the last access and RESET is lost! The command TDLINE fixes this problem by updating the internal clock each half second while displaying the time and date on the top of the display. Unfortunately this method must use the vertical blank interrupt which runs the risk of interfering with other user applications. The TIME command from older SpartaDOS versions is similar to TDLINE in its inherent problems.

If you have an R-Time 8 cartridge installed, SpartaDOS 3.2 will recognize it and use the R-Time 8 in place of the internal jiffy clock.

The new internal TIME and DATE commands are used to display and set the time and date. The new internal TD command is used to turn the time and date display line on and off (ie. TD ON or TD OFF) once TDLINE has been installed.

The old external time and date support commands (TD, TSET, XTD, TIME and SET) have not been changed. They may still be used with SpartaDOS 2.3 and 1.1 The reason for this will become apparent later.

Keyboard Buffer
A 32 byte keyboard buffer is now internal. The repeat rate has been doubled for faster operation. The keyboard buffer allows you to type ahead of the computer (up to 32 characters) even while disk I/O is occurring. You may disable this keyboard buffer by the KEY OFF command. Use KEY ON to turn the buffer back on.

Mini-buffering
Due to the nature of the CIO and the large database needed for SpartaDOS to operate, single GET and PUT operations tended to be slow. SpartaDOS 3.2 now contains mini-buffering that allows these operations to be performed faster. This also means faster INPUT and PRINT operations on disk files. Mini-buffering is not used with Atari DOS 2 formatted diskettes.

RESET
Now when you press RESET while in SpartaDOS with a cartridge present, control will remain with SpartaDOS. This also solves the conflict involving the warm start of BASIC that may have occurred under unusual situations. Optionally, AUTOBAT.COM can be used which causes a particular batch file to execute after pressing RESET.

AUTORUN.SYS and STARTUP.BAT
When SpartaDOS is booted, it will try to load an AUTORUN.SYS file. If successful, this file will run and then pass control to the cartridge. Otherwise, SpartaDOS will try starting a STARTUP.BAT file. If successful, control will pass to the Command Processor of SpartaDOS, otherwise, control will pass to the cartridge.

BASIC XE
The original reason for writing version 3.2 was to be compatible with BASIC XE. This has been done, but resulted in some major changes. First MEMLO had to be drastically increased (to just below Atari DOS 2). As long as MEMLO had to be increased, it may as well be pushed up near Atari DOS 2 with many extras

added.

Supra Hard Disk Interface
Support for the Supra Hard Disk Interface has been added, including the new BYPASS command which allows the use of floppy drives one and two, with the hard disks as three and four. This differs from the default set up - a 'fake' floppy as one, a real floppy as 2 and the hard disks as three and four.

DOS 2.5 Extended Read Capability
SpartaDOS version 3.2 can directly read all sectors on an Atari DOS 2.5 formatted diskette in enhanced density. This means you can boot SpartaDOS 3.2, insert a full Atari DOS 2.5 enhanced density diskette and RUN or COPY files from the directory which extend past sector 720. You still cannot write to this area of the diskette, however, read capability should be sufficient for most users.

Chapter 3___Commands Added to SpartaDOS 3.2

The following is a summary of the new commands added to SpartaDOS 3.2

TIME Command
Purpose - This command displays the current time and allows you to set the time

Syntax
TIME

Type and Restrictions
Internal under CP version 3.2
(this is totally different from the old external TIME command)

Remarks
This command produces the following output:

```
Current time is 12:34:56pm
Enter new time:
```

You may enter the new time or press <RETURN> if you don't want to set a new time. Enter the time in the format: 'hh:mm:ssx' where 'hh' is the hours, 'mm' is the minutes, 'ss' is the seconds and 'x' is an 'a' or a 'p' to distinguish between AM and PM.

NOTE: time is entered using a 12 hour clock rather than a 24 hour clock as in

the older commands.

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

Syntax
DATE

Type and Restrictions
Internal under CP version 3.2

Remarks
This command produces the following output:

```
Current date is 10/07/85
Enter new date:
```

You may enter the new date or press <RETURN> if you don't want to set the new date. Enter the date in the format: 'mm/dd/yy' where 'mm' is the month, 'dd' is the date and 'yy' is the year.

TD Command
Purpose - This command allows you to turn the time and date display line on and off

Syntax
TD ON or TD OFF

Type and Restrictions
Internal under CP version 3.2
(this is totally different from the old external TD command)

Remarks
The time and date display line is not supported internally by SpartaDOS. If you want the time and date display, you must first enter the TDLINE command to install the handler. This is explained further in the description of the time and date vectors.

TDLINE Command
Purpose - This command installs the time and date display line handler. The internal TD command controls this handler.

Syntax
TDLINE

Type and Restrictions
External under CP version 3.2
(this command may not be used with earlier versions)

Remarks
This command installs a handler into the system that gets accessed by the SpartaDOS time and date vectors. The functions TDON and FMTTD are provided by this handler. The TDON function causes this handler to patch itself into the vertical blank interrupt. During the interrupt, it calls the FMTTD function through SpartaDOS (returns the formatted time and date line), converts it to display format and then displays it on the expanded top line.

RTIME8 Command
Purpose - This command installs the routines needed to access the R-Time 8 cartridge for SpartaDOS versions 2.3 and 1.1 It is also used with Atari DOS as an AUTORUN.SYS file. RTIME8 contains the ZHAND and TDLINE functions.

Syntax
RTIME8

Type and Restrictions
External under CP versions 2.3, 1.1 and as AUTORUN.SYS with Atari DOS

Remarks
The RTIME8 installs the 'Z:' handler into the system. Under SpartaDOS 2.3 and

1.1, it updates TIMER and DATER (in COMTAB) for file time and date stamping. It is NOT to be used in conjunction with TD or XTD. It is provided for the 'Z:' handler compatibility.

NOTE: RTIME8 will also work with version 3.2 but it is preferable to use ZHAND and TDLINE instead. RTIME8 uses 178 more bytes than ZHAND and TDLINE combined. Also, ZHAND and TDLINE work with both the internal clock and the R-Time 8 cartridge under version 3.2 (the vectors are there). RTIME8 only supports the R-Time 8 cartridge but works with all versions of SpartaDOS.

ZHAND Command
Purpose - This command installs a 'Z:' handler that allows easy access to time and date functions from BASIC

Syntax
ZHAND

Type and Restrictions
External under CP version 3.2
(this may not be used with earlier versions)

Remarks
This command installs a device handler that provides an interface from a high level language to time and date functions. This handler converts requests from the CIO into calls to the SpartaDOS time and date vectors and visa versa. This may be used with either the R-Time 8 cartridge or the internal time and date in SpartaDOS. Further documentation on the 'Z:' handler is provided later in this document (pg130).

KEY Command
Purpose - This command enables or disables the internal 32 key keyboard buffer.

Syntax
KEY ON or KEY OFF

Type and Restrictions
Internal under CP version 3.2

Remarks
You will normally want the keyboard buffer on. It turns out to be quite handy. Use it for a while. In rare cases, the keyboard buffer may interfere with a software program (DDT - Dunion's Debugging Tool of the MAC/65 cart for one). If this happens, simply disable the buffer by the KEY OFF command.

RAMDISK Commands
Purpose - These commands install a Ramdisk device in place of a physical disk drive. Since these commands depend on specific hardware, the correct Ramdisk

command must be used or an error will result.

Syntax
RD Dn: [/NE]
RD260 Dn: [/N]

N - no format
E - extended XE memory banks reserved

Type and Restrictions
External under all CP versions - command must match hardware

Remarks
These new Ramdisks support hardware modifications which can be made to the 800XL, 1200XL and 130XE to provide large Ramdisks. Both commands format the Ramdisk automatically in the 128 byte sectors unless the '/N' parameter is used.

RD.COM supports the standard 130XE, the 130XE with 64K RAM upgrade (from Ron Boling) and the new 256K RAMBO XL upgrade for the 800XL and 1200XL computers. The RAMBO XL upgrade is available from ICD either installed or in kit form. RD.COM, used with the RAMBO XL, gives a 192K Ramdisk with holds a full double density disk! RAMBO XL also makes your 800XL or 1200 XL fully compatible with BASIC XE and other 130XE programs that take advantage of its extra memory.

The '/E' parameter (supported with RD.COM only) reserves the first 64K bank area for programs which use the extended memory of the 130XE (such as BASIC XE). For example, 'RD D2: /E' will install a 128K Ramdisk when using an 800XL modified with RAMBO XL.

RD260.COM SUPPORTS THE 800XL RAM upgrade to 256K as published in the September 1985 BYTE magazine by Claus Buchholz. It is used like RD.COM and gives a 192K Ramdisk! This upgrade banks the lower 32K of memory unlike RAMBO XL which banks memory from $4000 to $7FFF like the 130XE.

The '/N' parameter indicates that the Ramdisk is not to be formatted. This allows you to reboot the system and be able to retrieve that data that was in the Ramdisk.

CAUTION: you may not power down the computer to reboot, it must be done with a RESET that causes a reboot (cold start). This is accomplished by setting memory location $244 to a non-zero value and then pressing RESET. This is very useful for running a BBS from the Ramdisk. If the DOS crashes, you may be able to recover everything from the Ramdisk. RUN E477 will also cause a cold start with valid Ramdisk data left intact.

SCOPY Command
Purpose - This command is used to perform a straight sector copy, to compact an entire diskette into a file on another diskette or to expand a file into an entire diskette.

Syntax

SCOPY Dn:[[path>]sourcefname] [/UR] Dn:[[path>]destinationfname] [/UR]

U - US Doubler sector skew copied
R - Ramdisk identifier

Type and Restrictions
External under all CP versions

Remarks
SCOPY is a sector copier that has three modes of operation. It can do a disk to disk copy (like DUPDSK except that it formats the destination and it copies all sectors - this means it will copy Atari DOS 2 diskettes also). The other two modes of operation allow either the compaction of an entire diskette to a file on a destination diskette, or the creation of a destination diskette from a previously compacted file. This is an easy method for making multiple copies of the same diskette (non-copy protected) using a Ramdisk. In fact, this is how ICD creates its distribution diskettes.

The mode of operation is strictly determined by the syntax of the command. The first disk or file is always the source and the second disk or file is always the destination of the operation (as is true with the COPY command). The three modes of operation are basically:

```
disk to disk   - simple sector copy
disk to file   - compact diskette to file
file to disk   - expand file to diskette
```

If a file name follows the 'Dn:' then that item is considered a compacted 'file', - and with no filename, it is considered a 'disk'. A 'file to file' operation is ILLEGAL with SCOPY.

The destination diskette is automatically formatted in the same density as the source disk was. The compacted file stores density information from the original disk it was created from. It does not matter what density the file resides on. Skew may be changed between UltraSpeed (US Doubler) skew and standard Atari skew with the '/U' parameter described later.

CAUTION: for disk to file and file to disk copies, no prompt is given, the operation is performed without intervention. ALWAYS MAKE SURE YOUR DESTINATION DISK CAN BE FORMATTED. You will lose all previous information stored on it. For disk to disk copies, you are prompted for source and destination diskettes as required.

A 'disk' parameter may be followed by either a '/R' or '/U' parameter. The '/U' indicates that the diskette is US Doubler sector skew (for source) or is to be formatted in US Doubler sector skew (for destination). The '/R' parameter indicates that the diskette is really a Ramdisk. ONLY THE RD.COM RAMDISK WILL ACT AS A 'DISK' WITH SCOPY. RD.COM creates a Ramdisk which is a complete model of a true disk drive. It allows formatting, configuring, read configuration and status commands. The other Ramdisk handlers do not. Always use a 'space' as the delimiter between the device or filename and the slash '/' before the parameter.

When using SCOPY, the copy process is performed as efficiently as possible.

This means that if a drive contains a US Doubler and the diskette is not in UltraSpeed skew, SCOPY will adjust and optimize for fastest copy time. SCOPY will work with single, double and 1050 enhanced density diskettes. It is not intended for the specialty (eight inch or double sided) drives.

Example - disk to disk copy using a single US Doubler 1050 drive

```
SCOPY D1: /U D1: /U
```

This will cause an UltraSpeed sector skew diskette to be duplicated. Leave out the '/U' when copying standard Atari DOS 2 diskettes. For multiple disk duplication using a Ramdisk and a single drive, a typical pair of batch files would be:

```
RD D3:
COPY SCOPY.COM D3:
COPY BAT2.BAT D3:
D3:
;Insert diskette to be copied in drive 1
PAUSE
SCOPY D1: /U D3:MYDISK
-BAT2


;Insert destination diskette in drive 1
PAUSE
SCOPY D3:MYDISK D1: /U
-BAT2
```

The second group of commands would be in the batch file 'BAT2.BAT', the first group would be 'BAT1.BAT'. This sequence will use 'D1:' as source and destination while using 'D3:' as a temporary drive.

Note that SCOPY will NOT prompt in this sequence, thus the PAUSE command is used for the switching of diskettes in drive 1. You are prompted when performing drive to same drive duplication as in the first example.

For more information about the batch files and how they work, see Chapter 14 in the SpartaDOS Construction Set Owner's Manual.

BYPASS Command
Purpose - This is a special command for the Supra Hard Disk Interface to allow the use of a floppy drive 1.

Syntax
BYPASS

Type and Restrictions
External under CP version 3.2

Remarks
This command is similar to the Ramdisk commands in that it will knock out the

partitioned hard disk drive 1 and replace it with floppy drive 1 (assuming there is a physical floppy drive configured as 1). BYPASS patches itself into the SIO vector (of SpartaDOS) and checks for a drive 1 access. If so, then it passes control directly to the SpartaDOS serial I/O routines, thus the front end parallel I/O is skipped. BYPASS is needed if you wish to use a floppy drive as drive one rather than the hard disk acting as drive one.

CAUTION: IF USED FROM A BATCH FILE, MAKE SURE THAT THE BATCH FILE IS RUNNING FROM A DRIVE OTHER THAN DRIVE 1.

AUTOBAT Command
Purpose - AUTOBAT causes the specified batch file to be run whenever RESET is pressed.

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

Type and Restrictions
External under CP version 3.2

Remarks
This command patches itself into the SpartaDOS INIT vector (refer to the technical structure update) to cause control to be passed to the specified batch file after RESET is pressed. Refer to Chapter 14, 'Input/Output Redirection' in the SpartaDOS Construction Set Owner's Manual for information about batch files.

Chapter 4___Update on Technical Structure

Another version of SpartaDOS...why not? Actually, the major reason we decided to design another version of SpartaDOS was to achieve compatibility with BASIC XE which was released right after SpartaDOS version 2.3 came out. The actual problem was that BASIC XE was determined to use the memory at $C000 thru $CBFF and $D800 thru $DFFF which was already occupied by SpartaDOS 2.3 Since everybody seems to like BASIC XE (including us), it was decided that we would give up that area and move the code down to low memory. End of story? - no, of course not. As long as MEMLO had to be pushed up, we might as well start adding features... and add we did...

SpartaDOS Time/Date and other Vectors
SpartaDOS 3.2 contains many vectors pertaining to the setting, reading,

displaying of the time and date, executing command lines, initializing the system and many more. These vectors are contained in the RAM under the Operating System starting at address $FFC0. They may be accessed by the following instructions:

```
    LDA $D301      ;PIA
    PHA            ;save old value of port b on stack
```



```
    AND #$FE       ;set bit 0 to off
    STA #D301      ;enable RAM under the OS
    JSR VGETTD     ;call routine at $FFC0
    PLA
    STA $D301      ;restore port b (enable OS)
```

These functions each contain a jump (JMP) instruction to the appropriate function. If a function is not initially supported (as in TDON), the vector will contain a SEC and RTS instruction rather than a JMP. The following vectors are currently supported:

VGETTD $FFC0 This function returns the current time and date at COMTAB locations TIMER and DATER. On return, the carry flag is set if the function failed. When a file is opened for write, SpartaDOS makes a call here to update TIMER and DATER so it can move this data into the directory entry. Also the TIME and DATE internal commands make calls here to get the current time. TDLINE and ZHAND also use this vector.

VSETTD $FFC3 This function sets the time and date. On entry, the new time and date are at COMTAB locations TIMER and DATER. On return, the carry flag is set if the function failed. This vector is used by the commands TIME, DATE and the ZHAND set functions.

VTDON $FFC6 This function turns the time and date display line on or off. On entry the Y register contains zero to turn the line off, or one to turn the line on. On return, the carry flag is set if the function failed. This function is not supported internally by SpartaDOS. The TDLINE handler patches into this vector for use by the TD command along with the ZHAND XIO functions 38 and 39.

VFMTTD $FFC9 This function returns the formatted time and date line into a user supplied buffer. On entry the X and Y registers contain the high and low byte of the buffer address respectively. On exit, the carry flag is set if the function failed. This function is not supported internally by SpartaDOS. The TDLINE handler patches into this vector for use by TDLINE and ZHAND routines.

VINIZ $FFCC This vector is called after SpartaDOS has finished initializing itself after a RESET occurs. The command AUTOBAT patches into this vector to start a batch file right after RESET. By

initialization, we mean as a result of making a call through DOSINI - ie. JMP DOSINI. The OS monitor routine calls this vector before it enters a cartridge or DOS.

VINTZ2 $FFCF This vector is called after SpartaDOS has finished initializing itself after a NON-RESET occurs. Several SpartaDOS commands (such as SCOPY and UNERASE) will initialize SpartaDOS before and after they perform their function. They do this by jumping through the DOSINI vector as does the OS monitor routine after a RESET.

VXCOMLI $FFD2 This vector calls the Command Processor to execute a command line given at LBUF. BUFOFF should be zero on entry. Any errors that may occur as a result of executing the command line shall be printed as usual. No prompts are printed before or after command

execution. This is the method that a future DUP.SYS for SpartaDOS could use to perform its functions.

VCOMND $FFD5 This vector calls the main Command Processor program entry point. You may patch into this vector if you wish to supply your own Command Processor. The current one simply prints the prompt, inputs a line, calls VXCOMLI and jumps back to the beginning. This is the method DUP.SYS uses to gain entry from a DOS command in BASIC.

VPRINT $FFD8 This vector points to the SpartaDOS general print routine. The calling method is:

```
JSR VPRINT
DB 'This is a message',$9B,-1
```

VKEYON $FFDB This function turns the keyboard buffer on or off. On entry, the Y register contains zero to turn the buffer off, or a one to turn the buffer on. This function is supported internally by SpartaDOS.

Updates in the COMTAB Data Table

Several updates to COMTAB locations have been made. For general information about COMTAB, refer to the SpartaDOS Construction Set Owner's Manual.

DWARM [COMTAB-21] This location contains a copy of the WARMFLG (location 8) that will be used when a cartridge is entered, it indicates if user memory is valid and a zero indicates that a memory destructive command was entered (such as COPY) or that a binary file was loaded.

DDENT [COMTAB-19] This is the table of bytes per sector for each drive (1-8). A zero indicates 256 bytes per sector and a 128 indicates 128 bytes per sector.

WARMST [COMTAB-1] NOT USED

SBUFF [COMTAB+28] This is the start address of SpartaDOS sector buffers.

SMEMLO [COMTAB+30] This is the top of SpartaDOS low memory. Handlers added since boot take up the memory between SMEMLO and MEMLO.

INCOMND [COMTAB+32] A one here indicates that we are in the Command Processor (entering commands, etc.). A zero indicates that we are in BASIC or some cartridge program. This is used by the initialization routine to determine whether to enter the Command Processor or not.

SpartaDOS Version Identification

When writing new commands, it is necessary to distinguish between versions of SpartaDOS. From now on, two identification bytes at $700 and $701 are contained in all versions (from 2.5 on). Location $700 will always contain a $53 (S) and location $701 will contain the version number ($25, etc.).

Other Notes About Version 3.2

Since the SpartaDOS initialization now gives control directly to the Command Processor (CP) if INCOMND is true (-1), the CP may inadvertently be entered when a user program performs a warm RESET (through $E474 vector). To avoid this, a -1 placed at memory location $702 will disable the initialization routine from running the CP.

BASIC XE Notes

To be compatible with BASIC XE, SpartaDOS could not use memory from $C000 thru $CBFF and $D800 thru $DFFF. However, normally SpartaDOS uses these areas, the first area for both AINIT and the verbal error messages and the second for buffers. Thus, when using BASIC XE, both the AINIT command and verbal error messages are disabled, and eight buffers are allocated just below MEMLO (increasing MEMLO by $400 bytes). The eight buffers are a decrease from the sixteen buffers normally maintained at $D800 thru $DFFF.

Supra Hard Disk and Parallel Bus Notes

To be compatible with the Supra Hard Disk Interface and any future parallel bus devices, it was necessary to vacate $D800 thru $DFFF (as with BASIC XE). Parallel bus use is determined and automatically compensated for by checking PDVMSK ($247) to see if any parallel devices are on line. If a device is present, SpartaDOS uses eight buffers at low memory and does not use $D800 thru $DFFF (therefore increasing MEMLO by $400).

Chapter 5___The BASIC Time and Date 'Z:' Handler Functions

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

General Notes

It is assumed that a 'Z:' handler has been installed into your system. This is accomplished by the command 'ZHAND' under SpartaDOS 3.2 (also install the display with 'TDLINE' if using a display function under 3.2). If using an earlier version of SpartaDOS or Atari DOS 2 (or versions thereof), you must use the 'RTIME8.COM' file. Type 'RTIME8' if using a Command Processor driven DOS (like SpartaDOS), or rename the file to 'AUTORUN.SYS' and boot the diskette if using a DUP.SYS type of DOS (like Atari DOS 2.5).

Throughout these examples, 'IOCB' represents an Input/Output Control Block number from 1 through 7.

Turn Time and Date Display ON (TD ON)

Syntax
```
XIO 38,#IOCB,0,0,"Z:"
```


Notes
An extra line will appear at the top of the display containing the current time and date. The SpartaDOS version number will also be displayed. If you are using the 'RTIME8' handler for Atari DOS 2, the message 'R-Time 8' will appear instead of the SpartaDOS version message. An error 139 (Device NAK) will occur if the 'TDLINE' handler has not yet been installed under SpartaDOS.

Turn Time and Date Display Off (TD OFF)

Syntax
```
XIO 39,#IOCB,0,0,"Z:"
```

Notes
The extra line at the top of the display is removed. This has no effect on SpartaDOS's internal time keeping.

Read Formatted Date

Syntax
```
XIO 34,#IOCB,0,0,"Z:"
INPUT #IOCB,DATE$
```

Notes
This sequence will read the date into 'DATE$'. This string should be at least 13 characters long (as defined by the BASIC 'DIM' statement). The string will be returned in the format 'Mon 21-Oct-85'. Attempts to read more characters will return an end of file error 136. The IOCB must be opened for read prior to the execution of these statements (ie. OPEN #1,4,0,"Z:"). An error 139 (NAK) will be returned if the TDLINE handler has not yet been installed when using SpartaDOS 3.2

Read Formatted Time

Syntax
```
XIO 32,#IOCB,0,0,"Z:"
INPUT #IOCB,TIME$
```

Notes
This sequence will read the time into 'TIME$'. This string should be at least 10 characters long (as defined by the BASIC 'DIM' statement). The string will be returned in the format '10:20:46am'. Attempts to read more characters will return an end of file error 136. The IOCB must be opened for read prior to the execution of these statements (ie. OPEN #1,4,0,"Z:"). An error 138 (NAK) will be returned if the TDLINE handler has not yet been installed when using SpartaDOS 3.2

Read Unformatted Date (DATE)

Syntax
```
XIO 35,#IOCB,0,0,"Z:"
GET #IOCB,DAY : GET #IOCB,MONTH : GET #IOCB,YEAR
```

Notes
This sequence will read the numerical date into the variables, 'DAY', 'MONTH' and 'YEAR'. The year is returned as the last two digits of the actual year (ie. 1985 is returned as YEAR 85). Attempts to read more characters will return an end of file error 136. The IOCB must be opened for read prior to the

execution of these statements (ie. OPEN #1,4,0,"Z:"). No error will occur with this command under SpartaDOS 3.2 since this is an internal function of SpartaDOS 3.2

Read Unformatted Time (TIME)

Syntax
XIO 33,#IOCB,0,0,"Z:"
GET #IOCB,HOUR : GET #IOCB,MINUTE : GET #IOCB,SECOND

Notes
This sequence will read the numerical time into the variables, 'HOUR', 'MINUTE' and 'SECOND'. The hour is returned using the 24 hour clock where 0 is midnight, 12 is noon and 15 is 3PM. Attempts to read more characters will return an end of file error 136. The IOCB must be opened for read prior to the execution of these statements (ie. OPEN #1,4,0,"Z:"). No error will occur with this command under SpartaDOS 3.2 since this is an internal function of SpartaDOS 3.2

Set Date (DATE)

Syntax
XIO 37,#IOCB,0,0,"Z:"
PUT #IOCB,DAY : PUT #IOCB,MONTH : PUT #IOCB,YEAR

Notes
This sequence will set the date from the variables, 'DAY', 'MONTH' and 'YEAR'. The year is given as the last two digits of the actual year (ie. 1985 is given as YEAR 85). Attempts to write more characters will return an end of file error 136. The IOCB must be opened for write prior to the execution of these statements (ie. OPEN #1,8,0,"Z:"). An error 139 (NAK will occur if the handler is unable to set the date. The set function is an internal function of SpartaDOS 3.2

Set Time (TIME)

Syntax
XIO 36,#IOCB,0,0,"Z:"
PUT #IOCB,HOUR : PUT #IOCB,MINUTE : PUT #IOCB,SECOND

Notes

This sequence will set the time from the variables, 'HOUR', 'MINUTE' and 'SECOND'. The hour is given using the 24 hour clock where 0 is midnight, 12 is noon and 15 is 3PM. Attempts to write more characters will return an end of file error 136. The IOCB must be opened for write prior to the execution of these statements (ie. OPEN #1,8,0,"Z:"). An error 139 (NAK will occur if the handler is unable to set the time. The set function is an internal function of SpartaDOS 3.2

Some Examples Using the 'Z:' Handler

The following program will keep a constant time display:

```
10 DIM TIME$(13) : POKE 752,1 : REM turn cursor off
20 OPEN #1,4,0,"Z:"
30 XIO 32,#1,0,0,"Z:"
40 INPUT #1,TIME$
50 POSITION 2,0
60 PRINT TIME$
70 GOTO 30
```

Do not be alarmed if the time and date display line seems to stop while running this program. This is because both the TDLINE and the ZHAND routines try to use the SpartaDOS time and date vectors and priority is given to ZHAND rather than the interrupt display handling. If using the 'RTIME8' handler under Atari DOS 2 or SpartaDOS 2.3 and earlier versions, there is no conflict.

Chapter 6___Using the Supra Hard Disk Interface With SpartaDOS

Supra Corp. (formerly MPP) has released a parallel interface for hard disk drives (Winchester type). As previously promised, SpartaDOS is ready to be the DOS of choice for hard disk use. If you already have a Supra Hard Disk System up and running under another DOS, then skip the section on formatting and proceed to the section entitled 'Configuring for SpartaDOS'. If you are starting from scratch, then please read on.

Formatting the Hard Disk

The hard disk system stores important information about the type of hard drive being used on the first track of the disk. This information is written when the disk is formatted by the FORMWIM program. When the computer is powered up, the interface tries to read this information. If the information is not present (your drive has not been formatted before), the interface assumes that the hard disk is not usable and turns the computer system back to normal (ie. NO HARD DISK). Thus, there are two methods to format the hard disk, depending on 1) if the disk has already been formatted by FORMWIN (go to the 'Configuring for SpartaDOS' section) or 2) not (read on).

The first time the hard disk system is powered up the hard disk is totally blank. Since the drive information is not present, the interface will disable the hard disk and attempt to boot off of floppy drive 1. You must have a floppy drive set up as drive 1.

If this is NOT the first time (and the first track has not been trashed), the hard disk interface will attempt to boot from the 'fake' floppy on the hard disk drive. If the system still needs to be booted from the floppy disk (set up as drive 2), hold down the <HELP> key while powering up the computer. This will allow the system to recognize the hard disk drive, but still boot from floppy disk drive 2.

Before powering up the computer system, make sure there is a correct floppy disk drive attached and configured for drive 1 or drive 2 as per the information in the preceding paragraph. Turn on the hard disk sub-system and wait for it to come up to speed. Insert the SpartaDOS 3.2 master diskette into the correct disk drive. Remove all cartridges and hold down the <OPTION> key to disable BASIC. Turn the computer on (hold down the <HELP> key also if this is not the first time the system has been powered-up). If everything is correctly connected and turned on, the computer should boot from the floppy.

The first step in setting up the hard disk system is to format the hard disk. Use the external command 'FORMWIN'. This program will ask for drive number 0 or 1, enter 0. The next prompt will ask for drive type (0 through 8). Choose the drive type number from the following table that corresponds to the type of hard disk attached to the hard disk system.

NUMBER	SIZE	TYPE	SPECIFICATIONS
0	5 Meg	ST-506	153 Cyl/4 Hds
1	5 Meg	TM-501/ST-706	320 Cyl/2 Hds
2	10 Meg	TM-502/ST-712	320 Cyl/4 Hds
3	10 Meg	ST-506	230 Cyl/6 Hds
4	11 Meg	TM-503	306 Cyl/6 Hds

(Select '2' if you have a Supra Corp. Hard Disk System)

The format program will warn you that all data on the hard disk will be lost, answer 'Y'. The hard disk will be formatted and then verified for bad tracks. This process will take from 10 to 20 minutes.

When the format program is finished, turn only the computer OFF. Now attach a floppy drive configured as drive 2 and re-boot the computer while holding the <HELP> and <OPTION> keys down. Now you must configure the hard disks for SpartaDOS.

Configuring for SpartaDOS

First, use the XINIT command to initialize drive 1 and write SpartaDOS 3.2 to it (the 'fake' floppy). You should initialize the drive 1 portion of the hard disk as a 40 track, single sided, double density drive. Answer 'N' to UltraSpeed sector skew.

Next use the HDINIT command to initialize drive 3 (the large portion of the

hard disk). All that is required is a volume name and the drive number (3). Drive 4 will be initialized if you are using a second hard disk drive connected to your XEBEC controller board.

Now the hard disk is totally configured and you should never have to initialize it again. Now re-boot the computer, and simply watch the speed as SpartaDOS loads in.

Using the Hard Disk

The hard disk sub-system (the physical drive unit) is the only sensitive part of a hard disk system. When the hard disk sub-system is running it should never be moved or bumped. The heads in the hard disk are extremely close to the surface of the hard disk and any movement may cause the heads to hit the media surface causing a literal head crash.

The hard disk system should be turned on in a specific sequence. First turn on the hard disk sub-system and wait for it to come up to speed (the motor sound should be steady). If the computer is turned on before the hard disk is ready, the computer will either try to boot off of floppy drive 1 or give 'BOOT ERROR'. To boot the computer from a floppy disk, put the diskette into disk drive 2 and hold down the <HELP> key while turning the computer on.

The following is a description of the utility programs by SUPRA CORPORATION for use with the hard disk.

FORMWIN	This program formats and verifies the whole hard disk.
LOCK1	This program write protects the 'fake' floppy drive.
OPEN1	This program unprotects the 'fake' floppy drive.
LOCK34	This program write protects hard disk drives 3 and 4.
OPEN34	This program unprotects hard disk drives 3 and 4.
ADDLF	This program adds line feeds to all carriage returns that are issued to the parallel printer port on the interface.
RMVLF	This program stops the adding of line feeds to carriage returns issued to the parallel printer port.
PARK	This program positions the heads at the innermost track. A good practice before moving the hard disk system.

Some Notes About SpartaDOS

You may also use the SpartaDOS commands LOCK, UNLOCK, PROTECT and UNPROTECT, to prevent accidental erasure of your files on the hard disk system. The BYPASS command from SpartaDOS allows the use of a floppy as drive 1. SpartaDOS will support 128 files per directory with an unlimited number of directories, and up to 16 megabytes as one drive. Use and enjoy.