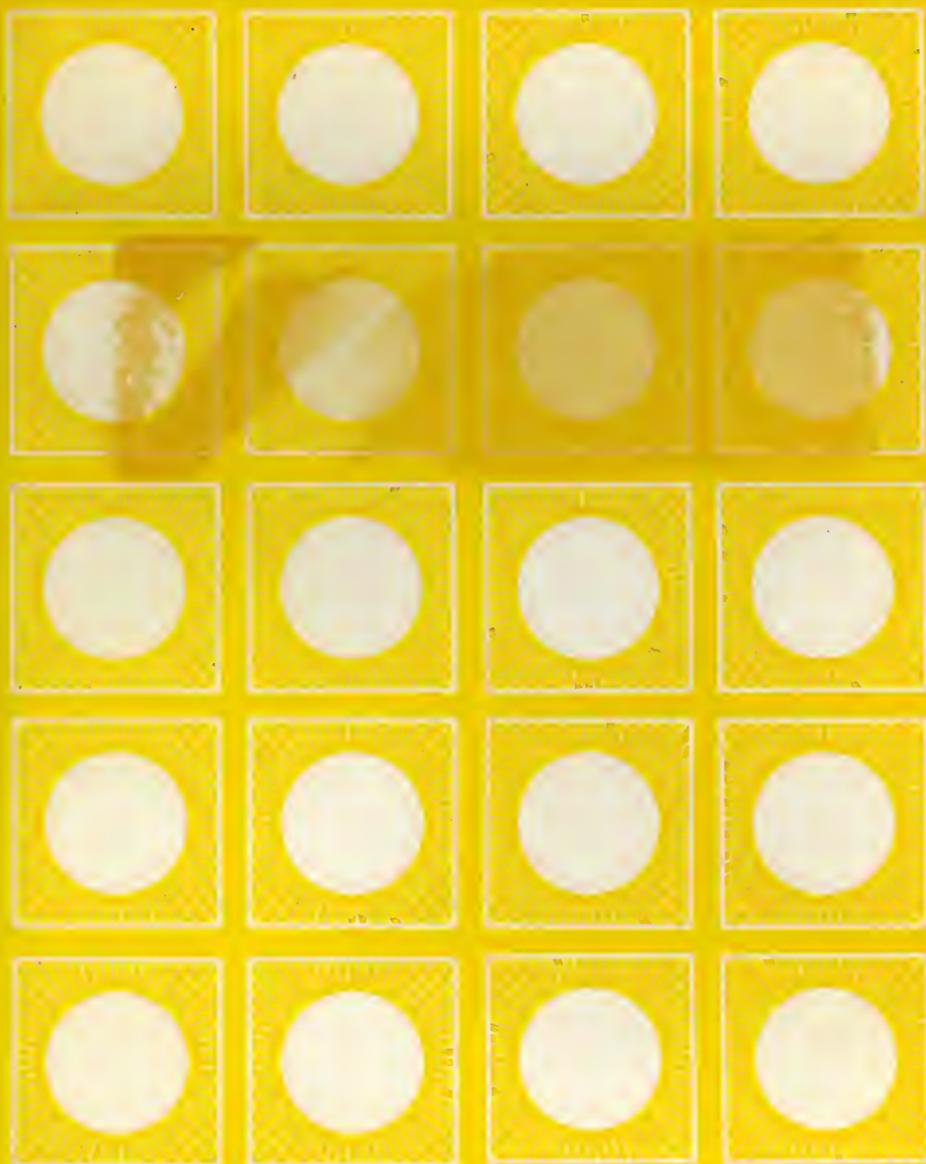


Quite simply the best DOS
ever written for Atari computers.

TOP-DOS™



TOP-DOS MANUAL

MAY 1984

COPYRIGHT 1984 R.K.BENNETT

ATARI, ATARI 810 Disk Drive and ATARI 850 Interface Module are trademarks of Atari, Inc., which are used in this manual.
PERCOM is a trademark of PERCOM DATA Corporation.
OS/A+ DOS-XL are trademarks of O.S.S. INC.
TOP-DOS and DOS-MOD are trademarks of ECLIPSE SOFTWARE.
AXLON is a trademark of AXLON, Inc.
MOAIC and SELECT are trademarks of MOAIC Electronics.

COPYRIGHT

The TOP-DOS program and the contents of this manual are copyrighted materials. You may make as many copies of these files as you need for your own use, but PLEASE ask your friends to purchase their own copies! Annotate any disks to which you have written TOP-DOS with the label (c) 1984 R.K.Bennett.

WARRANTY

ECLIPSE SOFTWARE has made every effort to create a perfect product. However, TOP-DOS, like any software, may contain bugs or undesirable behavior, under conditions for which it has not been tested. You have 30 days to test this product and may return it within this period for full refund, if you do not find it satisfactory for your needs.

Furthermore, ECLIPSE SOFTWARE warrants the diskette to be free of defects for a period of 90 days, during which the diskette will be replaced free of charge. Beyond this period, replacement of a damaged or defective diskette will be made to the registered owner for a nominal charge to cover costs.

ECLIPSE SOFTWARE's responsibility is limited to this warranty and cannot be held liable for consequential damages.

TABLE OF CONTENTS

Chapter One: INTRODUCTION	1
The Product	1
Copyright	1
How to Use this Manual	1
Chapter Two: GETTING STARTED	3
System Requirements	3
OSA or OSB-ROM?	3
Single or Double Density	3
Boot Up	3
Use of the Screen Editor	4
Entering a Command	5
Help Facility	6
Errors	6
One-Line Commands	6
Chapter Three: OPERATING TOP-DOS	7
Interpreting the Menu Display	7
Heading Line	7
Status Display	7
Command Menu Display	8
Requesting Menu Display	8
Command Prompt	9
Menu Scroll	9
BREAK Key	9
SYSTEM RESET	9
Making a Backup Disk	9
Chapter Four: DEFINITION OF TERMS	
Filespec	10
Wildcards	11
Defaults	12
System Drive	12
Command Syntax	12
Brief Glossary	13

Chapter Five: TOP-DOS MENU COMMANDS	14
Common Command Notes	14
A - Directory Listing	15
B - Run Cartridge	19
C - Copy Files	20
D - Delete Files	24
E - Rename Files	25
F - Lock Files	27
G - Unlock Files	28
H - Write DOS Files	29
I - Initialize Disk	30
J - Duplicate Disk	33
K - Binary Save	34
L - Binary Load	36
M - Run at Address	38
N - MEM.SAV	39
O - Duplicate Files	40
P - Run Program	41
Q - Command File	42
R - Read/Store Memory	46
S - Set/Status	49
T - Trouble Report	53
U - Undelete Files	54
Chapter Six: FEATURES AND OPTIONS	56
Autorun File Facility	56
RS232 Autorun File	56
HELLO Facility	57
DOS-Resident Feature	57
Auto-RS232 Feature	58
Appendix A: STRUCTURE OF TOP-DOS	59
Appendix B: THE MEM.SAV FACILITY	63
Appendix C: HEXADECIMAL NOTATION	68
Appendix D: TOPICS FOR ADVANCED USERS	71
Appendix E: ERROR CODES	74
Appendix F: RAMDISK ENHANCEMENT	77

CHAPTER ONE INTRODUCTION

Welcome to TOP-OOS!

THE PRODUCT

TOP-OOS was created by Richard K. Bennett of ECLIPSE SOFTWARE as a result of experience with its predecessor, DOS-MOO. With all its professional features, TOP-DOS occupies the same space in memory as the ATARI OOS 2.0. Thus, TOP-OOS should be compatible with all your ATARI software.

A OOS (Disk Operating System) is the set of instructions that organizes the disk into files and allows the user to access them. A DOS allows you to:

- o Store program files on a disk
- o Retrieve stored programs
- o Create, Append, Copy and Delete files
- o Load and Save binary files
- o Transfer files to or from RAM, screen, disk, and printer.

In addition, with TOP-DOS you can:

- o Enter commands with a single line
- o Create powerful command files made up of a series of operations
- o Examine and change bytes in memory without the hassle of PEEK, POKE and PRINT
- o Alphabetize and compress your File Directory
- o Restore unintentionally deleted files
- o Call for HELP with a single keystroke!

HOW TO USE THIS MANUAL

TOP-OOS has many sophisticated features and facilities, not ordinarily found in a OOS for a home computer. Most of this manual is devoted to describing how to make the most effective use of these features. But do not be intimidated by the size of this manual. It has been carefully designed to allow you to begin using TOP-OOS almost immediately, without reading the whole manual.

If you are familiar with the ATARI DOS 2.0, you can begin using TOP-OOS without reading further, since it includes all of the ATARI OOS commands and capabilities.

If you are not familiar with the ATARI DOS, then you should read Chapters Two, Three, and Four. You can then use Chapter Five's command descriptions to refer to individual commands as you are learning them.

Some of the commands you will need to get started include:

- A - Get a Directory Listing of the files on a disk.
- I - Initialize a new disk before writing to it.
- H - Write the TOP-DOS files to a disk.
- J - Duplicate the files from one disk to another.
- T - Get a Trouble report, in case you get an Error Number.
- B - Go to BASIC (or other) cartridge.

If new to your computer or new to disk drives, be sure you have read your Atari operator's manual and your disk drive operator's manual.

Chapter Two: GETTING STARTED gives the system requirements, boot-up procedures, and enough of the basic features of TOP-DOS to allow the eager user to begin immediately.

Chapter Three: OPERATING TOP-DOS supplies information common to all command operations and tips on optimum use of TOP-DOS.

Chapter Four: DEFINITION OF TERMS defines terms used in this manual.

Chapter Five: TOP-DOS MENU COMMANDS describes each of the Menu Commands. TOP-DOS includes improved versions of all the ATARI DOS-2 commands, plus six new commands.

Chapter Six: FEATURES AND OPTIONS provides technically more complex information on some of the features and options of TOP-DOS, including the Autorun File facility, the RS232 Autorun file, the Hello file facility, the DOS-Resident feature, and the Auto-RS232 feature.

Appendix A: STRUCTURE OF TOP-DOS describes the structure of TOP-DOS on disk and in memory. It includes a memory map of TOP-DOS and discussions of its programs, storage, and buffers.

Appendix B: THE MEM.SAV FACILITY describes the method whereby TOP-DOS and user programs can share memory.

Appendix C: HEXADECIMAL NOTATION gives an introduction to the "hex" number system, together with a conversion table.

Appendix D: TOPICS FOR ADVANCED USERS describes some memory locations and subroutines available to advanced users.

Appendix E: ERROR CODES lists the error codes and messages which you may get and gives a brief explanation of their possible cause.

Appendix F: RAMDISK ENHANCEMENT describes how TOP-DOS can be enhanced to accommodate a "ramdisk" with the AXLON or MOSAIC RAM's.

CHAPTER TWO
GETTING STARTED

PRELIMINARIES

SYSTEM REQUIREMENTS

TDP-DDS is designed for your Atari home computer and up to eight disk drives. To attach your computer and drives, be sure to follow the instructions in your operator's manuals. You must have at least 32K of RAM to accommodate TOP-DDS.

Disk drives must be numbered so they can be addressed by the computer. A drive is numbered by setting the drive code switches, usually found on the back of the drive. Put a number label on the front of each drive, too, to remind you of its number.

OSA or DSB-ROM?

The Operating System (OS) for the ATARI computers was updated in 1982. Therefore some early 400 and 800 computers may have the outdated OSA-ROM. If this is what your machine has, a warning message will show up on your screen at boot, telling you that TDP-DDS will only work with DSB-ROM. Contact your Atari dealer about upgrading your system.

SINGLE OR DOUBLE DENSITY?

TDP-DDS works with single-density, double-density, and double-sided double-density drives. When you boot-up, a status display will tell you which drives you have online and their density.

BOOT UP

Having done all the preparations noted above,

TURN ON your disk Drive-1. Wait until the busy light goes out, then

INSERT the TDP-DDS disk and close door, GENTLY.

Turn the computer ON to boot-up TDP-DDS. The busy light on your disk drive should come on.

The boot-up procedure loads the TOP-DDS memory-resident file (DDS.SYS) into the computer's memory. This process, which takes about eight seconds, prepares your computer to handle disk operations.

If the speaker on your TV or monitor is on, you will hear the sound of the data being transferred from the disk to your computer. This sound will become familiar to you, telling you that a disk transfer is proceeding normally.

If you have the BASIC cartridge inserted, "READY" will appear on the screen. With the ASSEMBLER cartridge, "EDIT" will appear. In either case type "DOS<RETURN>" to proceed into TDP-DDS with its Status and Menu Display.

If there is no cartridge, you will not get the READY or EDIT prompt but will proceed directly to TOP-DOS.

At this point, the file DUP.SYS (the Disk Utility Program which handles the Menu commands) will be loading into your computer's memory and the following message will appear on your screen to inform you of what is going on:

```
Read D1:DUP.SYS
```

At the end of this read operation, which takes about ten seconds, you will see something like this:

```
(C) 1984 R.K. BENNETT TOP-DOS v1.1

<I>S 2D 8=#BUFS
2000-BCIF=MEMLO-MEMTOP AUTO-RS232<ON>
MEM.SAV<OFF> DOS-Resident<OFF>
Bypass-Cartridge<OFF> Verify-Write<OFF>

<A> Directory <H> Writ DOS <O> Dup File
<B> Run Cart <I> Init Dsk <P> Run Prog
<C> Copy File <J> Dup Disk <Q> Cmd. File
<D> Delete <K> Bin Save <R> Read/Store
<E> rName <L> Bin Load <S> Set/Status
<F> Lock File <M> Run&Load <T> Trouble
<G> Unlock <N> MEM.SAV <U> Undelete
```

```
Type Command or ";" for Menu
```



This display is interpreted in Chapter Three.

The menu commands are now ready to be used! But first there are a few things you will need before you are ready to execute your first command.

USE OF THE SCREEN EDITOR

See your ATARI Operator's Manual for a full explanation of the uses of the Screen Editor. Atari's Screen Editor has some remarkable capabilities that are not available even on large computers. It is worth spending a little time to thoroughly learn how to use all of its features. They will come in handy as you use TOP-DOS, as well as, BASIC and other user programs.

Here are some reminders of what you can do:

The cursor control keys can be used to position the cursor anywhere on the screen. Insert and delete editing operations can be done on the screen before the final version of a line is entered by typing <RETURN>.

<CTRL>-1 will stop the screen from scrolling (useful in long lists).

<CTRL>-3 is the code for end-of-file, which is used to close a file that is created on the screen.

With the Screen Editor you can edit a line without retyping the whole line, or you can repeat an earlier command line simply by positioning the cursor anywhere on that line and hitting <RETURN>. Remember, in screen editing, nothing happens in memory until a <RETURN> is entered!

The Screen Editor can be used to correct lines in command files, to change bytes in memory via the R command, and even to edit short text files (20 lines or less) right on the screen without resorting to a word processor! (See the NOTES to the C Command for instructions on how to do this.)

ENTERING A COMMAND

The TOP-DOS Command Prompt ♥ is displayed when an operation is complete and TOP-DOS is ready to accept the next command. When you have a ♥, you may enter a command by typing in the Command Letter <RETURN>.

TOP-DOS will respond with a prompt message, indicating what parameters are required for this command. (The term "parameter" is explained in Chapter Four.)

For example:

```
♥C<RETURN>
COPY: from,to/A/M/N/Q/U
```

The ♥ is the Command Prompt from TOP-DOS. The "C" is the Command Letter that you type for the Copy command.

TOP-DOS responds to the command with the second line, which is called the "Parameter Prompt".

At this point you have a choice. You can either:

- o Abort the command by hitting the <BREAK> Key,
- o Type in parameters to complete the specification of the command, or
- o Just type <RETURN> to let TOP-DOS supply "default", parameters (available for most commands).

Characters which are not "legal" Command Letters are just ignored -- nothing happens -- the prompt ♥ returns.

IMPORTANT NOTE: The <RETURN> key must always be struck after typing each line. This is necessary even if you are typing only one character, as in this case of selecting a Menu command. The use of <RETURN> to enter input is consistent with the ATARI OS Screen Editor. (The use of the <RETURN> Key is indicated throughout this manual by <RETURN>.)

This procedure allows you to edit any mistakes before entering your input to the computer. Some other systems make the keyboard "live". With these systems, as soon as you type a character, the computer accepts it. You have no opportunity to make changes in your input before the computer uses it.

HELP FACILITY

TOP-DOS includes a friendly HELP facility. After any Menu Command Letter, just type "?<RETURN>" to get an explanation of that command. For example:

```
♥C?<RETURN>
```

will display an explanation of the C command. The HELP facility is also available after you have typed a Command Letter <RETURN> and the Parameter Prompt for the command is displayed. Again, just type "?<RETURN>" to get a HELP display for the command. For example:

```
♥C<RETURN>  
COPY: from,to/A/M/N/Q/U  
?<RETURN>
```

will display the desired information.

This HELP facility may also be used after a Command Prompt ♥ to give you general information about getting the Menu and entering a command.

NOTE: The HELP facility uses files with a ### "file extension". (For example, the HELP file for the "A" command is "A.###".) These files must be included on the disk in the "System Drive", for this facility to be active. If you request HELP for a command for which the corresponding HELP file is not on the System Drive, "ERR 170" (File not found) will be displayed. (See Chapter Four for the definition of "file extension" and "System Drive".)

ERRORS

Errors detected by TOP-DOS are handled in different ways. Some errors are considered innocuous -- such as an attempt to copy or delete a non-existent file -- and are simply ignored. Other errors result in a simple message, indicating the nature of the error.

Some errors, such as those originating from the Operating System (OS), are reported by display of an Error number. For example, the OS "Invalid Command" error is displayed as "Err 134". Error numbers can be interpreted using the T command, as described in Chapter Five.

ONE-LINE COMMANDS

After you have become familiar with the commands, you will no longer need the parameter prompts. You can then progress to entering a whole command on a single line. This is faster and uses much less space on the screen. Using less space will give you more work history on the screen, which you will find to be very helpful.

To enter a one-line command, type the Command Letter, followed by a space and then the parameters. For example:

```
♥C FILE1,FILE2<RETURN>
```

CHAPTER THREE OPERATING TOP-DOS

INTERPRETING THE MENU DISPLAY

What does the information on the screen at boot tell you?

Let's take a closer look at the Menu Display:

```
(c) 1984 R.K. BENNETT TOP-DOS v1.1

<I>S 2D 8=#BUFFS
2000-BC1F=MEMLO-MEMTOP AUTO-RS232<ON>
MEM.SAV<OFF> DOS-Resident<OFF>
Bypass-Cartridge<OFF> Verify-Write<OFF>

<A> Directory  <H> Writ DOS  <O> Dup File
<B> Run Cart  <I> init Dsk  <P> Run Prog
<C> Copy File  <J> Dup Disk  <Q> Cmd. File
<D> Delete    <K> Bin Save  <R> Read/Store
<E> rEname    <L> Bin Load  <S> Set/Status
<F> Lock File <M> Run&Load  <T> Trouble
<G> Unlock    <N> MEM.SAV  <U> Undelete
```

Type Command or "I" for Menu



HEADING LINE

The heading line gives the copyright notice and the current version number of TOP-DOS.

STATUS DISPLAY

The four-line Status Display which follows the heading line tells you:

i. On the first line, the drive numbers in the "drive list" and the density of each drive: S=Single-density, D=Double-density, Q="Quad"-density (double-sided double-density). No letter after a drive number signifies that the drive is off-line. The highlighted number indicates which drive is the "System" Drive (see Chapter Four).

Thus, in the above Status Display, Drive-i is single-density and is the System Drive, and Drive-2 is double-density.

2. Also on the first line of the Status Display, the number of buffers available for open files. Double-density drives require two buffers for each open file. Normally, three or four open files are sufficient for most applications programs.

The above Status Display shows 8 buffers for open files. This is sufficient for 4 open double-density files or 8 open single-density files, or some combination like 2 single-density and 3 double-density files, etc. (See Appendix A for more information on buffers.)

3. The second line of the Status Display gives you the bounds of the memory space available for user programs. The numbers are the addresses found in MEMLO and MEMTOP, in hexadecimal notation (hex), which is the established numerical system used to refer to computer memory locations and bytes.

NOTE: MEMLO and MEMTOP are memory locations set by the Operating System (OS) and other systems programs, including TOP-DOS. These and other special memory locations are listed in "Mapping the ATARI" by Ian Chadwick, published by COMPUTE! Books, a book you might find useful if you want to delve more deeply into the inner workings of your ATARI computer.

In the above display, the area between memory locations 2000 and BC1F (hex) is available for user programs.

4. The remainder of the Status Display consists of the status of the TOP-DOS features: Auto-RS232, MEM.SAV, DOS-Resident, which are described in Chapter Six, and Bypass-Cartridge and Verify-Write, which are handled in the S command in Chapter Five.

NOTE: To get a Status Display without re-booting, use the S command with the J (Just Status Display) parameter:

♥S J<RETURN>

COMMAND MENU DISPLAY

The Menu Display contains the Command Letters A through U and brief descriptions of the TOP-DOS commands. These commands are fully described in Chapter Five.

REQUESTING MENU DISPLAY

The next to last line of the above display reminds you to type ";" to have the Command Menu redisplayed whenever you want it:

⚡;<RETURN>

NOTE: This use of the semicolon ";" to get the Menu instead of the more commonly-used RETURN prevents the inadvertent redisplay of the Menu and the subsequent loss of work history on the screen.

COMMAND PROMPT CHARACTER

The last line of the above display shows the TOP-DOS Command Prompt ♥ and the cursor ■. The prompt is displayed whenever an operation is complete and TOP-DOS is ready to accept the next command from you.

MENU SCROLL

As you type in commands, the TOP-DOS Menu scrolls off the top of the screen, providing you with more work area. You can, of course, get the Menu back anytime by typing ";"<RETURN>".

BREAK KEY

The <BREAK> Key can be used to abort any command, if pressed when user input is expected, such as when TOP-DOS is waiting for command parameters.

The <BREAK> Key can also be used to interrupt a C Copy, D Delete, or E Rename command operation while in progress. This valuable capability allows you to terminate an operation without losing files.

SYSTEM RESET

If the system does not respond properly to your keyboard entry, and if the BREAK Key does not help, try the SYSTEM RESET Button. Its effect is similar to turning the computer OFF and ON to re-boot, except that program memory is not destroyed. (The 6502 microprocessor chip used in ATARI computers will "hang up" if it encounters an illegal instruction. If this happens, System Reset will not work, and you will be forced to re-boot to recover.)

MAKING A BACKUP DISK

You may wish to make a backup copy of your TOP-DOS disk as a safety measure in case it may become damaged.

First Format a disk, using the I command. Then use the J command to duplicate the TOP-DOS disk to the formatted disk. (See Chapter Five for descriptions of the I and J commands.)

CHAPTER FOUR DEFINITION OF TERMS

FILESPEC (File Specification)

A "filespec" is a string of characters which specifies the source or destination of a file. (A file refers to a display on the screen or a listing on the printer or a file on one of your disk drives.)

A filespec consists of a "Device Designator", a "Filename", and a "File Extension".

For example,

D1:CAMU.BAS

is a typical filespec of a disk file, and will be referred to throughout this section.

Device Designators: "P:", "E:", "Dn:"

The devices which can be used with the Menu commands are the Printer, the Screen, or a Disk Drive.

The Printer is designated by "P:". The Screen is designated by "E:" ("E", because ATARI uses the name "Screen Editor"). Disk Drives are designated by "Dn:" ("n" denotes the disk drive number, 1 through 8). "D1:" in the above example designates Disk Drive-1 as the Device.

If the device is either the "E:" or "P:", the filename and extension are inappropriate.

Filename and File Extension

A filename is the name of a file on a disk. It consists of 0 to 8 characters. A zero-length (null) filename is permitted, but not generally used.

The filename specified in the above example is "CAMU".

A file extension is a modifier of the Filename. It consists of 0 to 3 characters and is separated from the filename by a Period ".". A zero-length file extension is permitted and is frequently used.

A file extension is generally used to indicate the "kind" of file, such as SYS (system), TXT (text), BAS (Basic), TMP (Temporary), etc. But any "legal" characters will do.

In the above example, the file extension is "BAS".

NOTE: Throughout this manual, files with TXT extensions will be referred to as .TXT files.

If no file extension is specified after a filename, TOP-DOS presumes the extension to be null. If an extension is specified, it must be preceded by a period ".", whether or not the filename is specified.

Legal Characters for Filenames & Extensions

Any combination of the following characters, in any order, are legal characters for filenames and file extensions in TOP-DOS:

Upper case letters: A through Z

Lower case letters: a through z

Numerals: 0 through 9

Underscore: -

Number Sign: #

Dollar Sign: \$

Percent Sign: %

Ampersand: &

NOTE: Although TOP-DOS allows this flexibility in filenames, some programs -- such as the ATARI MEDIT -- restrict you to numerals and Upper-case letters (the character set of the ATARI DOS). Other programs -- such as ATARI BASIC -- do not have this restriction, so you can use the full TOP-DOS set.

ALSO: TOP-DOS intentionally does not allow ALL characters in filenames, which would have small value and could create problems for future extensions of TOP-DOS.

WILDCARDS

When searching for files on a disk, or when copying files, you may want to specify a particular set or class of files. This is accomplished the using the special characters "X" and "?" in the filespec. These characters are called "wildcards" because of their similarity with wildcards in card games, wherein a wildcard may be used for any card.

These wildcards can be substituted for characters in a filename and/or a file extension. Each "?" substitutes for a single character, while "X" substitutes for any string of characters.

Thus,

X.TXT

specifies any and all files with a TXT file extension, and

ABC.??T

specifies all files with the filename ABC and a three-character file extension ending in T.

Examples of legal filespecs, including some with wildcards:

D1:JOE.TXT Just this one particular file on Drive-1

D1:JOE\$xx.###	Shows the use of non-alphanumeric characters
D2:X.TXT	All .TXT files on Drive-2
D\$:X.TXT	All .TXT files on all drives
D1:tX	All files starting with "t" with a null extension on Drive-1
D1:JOE.X	All files with the filename JOE and any extension on Drive-1

DEFAULTS

The term "default" refers to a parameter or part of a parameter that TOP-DOS will supply when it is omitted from a command.

For example, for most commands the default filespec is "D1:X.X". Also, if the filename and/or extension is given, but the device designator omitted, the default device is the "System Drive".

The use of defaults makes your task of entering commands easier, because TOP-DOS supplies frequently used parameters -- or parts of parameters -- for you.

SYSTEM DRIVE

The System Drive -- normally Drive-1 -- is the drive which holds the "system" files. It also serves as the default drive number in a filespec. (The system files include DOS.SYS, DUP.SYS, MEM.SAV, and the HELP files.)

NOTE: Drive-1 is always used for the boot operation.

COMMAND SYNTAX

The term "syntax" refers to the specific format of a command. It specifies precisely how to structure a correct command.

For example, to get a Directory Listing, you would type the command:

ⓂA<RETURN>

TOP-DOS responds with the following "prompt" line, indicating the "parameters" required to complete this command:

DIR: filespec/n/A/B/C/D,list-file/N

A parameter is a string of characters that supplies additional information to a command. Parameters are separated by commas "," (No spaces, please!). If a parameter is omitted, its comma is still needed if a following parameter is given. In the above example for the A command, to omit the first parameter and give the second, you would type ",DIRLIST<RETURN>", to specify DIRLIST as the list-file.

The specification of the order and type of parameters constitutes the syntax of the command. The syntax for each command is given in Chapter Five.

A BRIEF GLOSSARY

Here are definitions of some other terms you will run across in this manual:

Binary Load File: A file containing an exact copy of the binary data in an area (or areas) of memory, together with the memory addresses of those areas.

Buffer: A storage area located in RAM used to temporarily hold data intended for further processing.

File: A named collection of related data.

File Directory: A list of the files contained on a disk. The File Directory is itself stored on the disk and includes other information about the files, such as the number of sectors they use.

FMS (File Management System): That part of TOP-DOS which includes the handlers for accessing disk files from user programs.

Program Memory: The area of memory available for user programs, as opposed to the area of memory used by the OS and DOS.

RAM (Random Access Memory): Memory in a computer that can be both read from and written to, as opposed to Read Only Memory (ROM), or as opposed to Disk Memory. RAM is generally volatile, that is, data contained in it is lost when the computer is turned off.

ROM (Read Only Memory): Memory which can only be read and cannot be written to. ROM is used to contain permanent programs, such as an OS or BASIC, which are not lost when the computer is turned off.

6502: The microprocessor computer chip used in the ATARI computers.

Sector: A portion of a track on a disk used to hold data.

Track: A ring of data on a disk which is divided into sectors.

<TOPDOS>-TYPE DISK: ECLIPSE SOFTWARE has created a file structure slightly different from ATARI'S. It accomodates the larger number of sectors required by the double sided disk drive and permits the alphabetization of File Directories. (This difference is the elimination of the file numbers from the sector links.) Disks with this file structure are referred to as <TOPDOS> type disks. Disks with the ATARI file structure are referred to as <ATARI>-type disks. TOP-DOS can read and write both type disks. (See the I command for further information.)

VTOC (Volume Table of Contents): The directory on a disk of the sectors which are available for use.

Write-Protection: A method of preventing a disk drive from writing to a disk. Accomplished on 5-1/4" diskettes by fastening a sticker over the notch on the disk.

CHAPTER FIVE
TDP-DOS MENU COMMANDS

For each command letter, the following information is provided:

- o The Command Letter and the Title of the command
- o The SYNTAX of the command
Brackets [] are used to indicate input that is optional.
Vertical Bar "|" indicates a choice.
Options are preceded by a Slash "/", which must be included before the option character, if the option is selected.
Commas "," separate parameters and must be included even when the parameter is omitted, if a following parameter is given.
Spaces are NDT permitted between parameters.
- o The PURPOSE of the command
- o A DESCRIPTION of how the command operates
- o The SYNTACTIC ELEMENTS
Brackets [] are used to show the default value, if one exists. (This use of Brackets is related to their use in the SYNTAX of the command, where they are used to indicate optional elements.)
- o EXAMPLES

COMMON COMMAND NOTES

The following notes apply throughout this chapter. Refer to these notes as you read the command descriptions.

DEFAULT DRIVE -- For simplicity of explanation, the default drive is referred to in this chapter as Drive-I. More precisely, the default drive is the System Drive -- which is normally Drive-I, but may be changed by the S command.

PROGRAM MEMORY USED AS BUFFER -- The C, J, and D commands use program memory as buffer in the copying operations, and the A command uses it for sorting the File Directory if the /A or /C options are selected.

The J and D commands use ALL of program memory from the end of TDP-DOS through MEMTOP. The C and A commands use only the top 1020 (hex) bytes of program memory just below MEMTOP.

The use of this program memory will of course destroy any program or data you may have there. Furthermore, since BASIC uses all of memory between MEMLD and MEMTOP, any use of program memory invalidates any BASIC program you may have stored in its buffer. Therefore, if you intend to use any of the above commands, SAVE your BASIC program before typing "DDS" to go to the TDP-DOS Menu commands.

ATTEMPTING TO WRITE TO A WRITE-PROTECTED DISK -- You will get an "ERR 144" message if you attempt to write to a disk which is write-protected, that is, which has an open notch on its side. Disk writes may occur with the A, C, D, E, F, G, H, I, J, K, O, and U commands.

SYNTAX -- A [filespec][/n:A|C|D|B|I[,list-filespec[/N]]

PURPOSE -- This command is used to get a listing of the files in the File Directory of a disk.

DESCRIPTION -- The listing gives the names of files on the disk, in the order that they appear in the Directory, together with the number of sectors in each file.

The status of each file is indicated by these initial characters:

```

"+" File is locked
"- " File is deleted
"! " File was left open (not properly closed)

```

At the end of the directory listing, a summary is given showing the number of sectors and the number of files listed, together with the number of sectors and the number of files which are free on the disk, available for use. Also on the summary line are the disk "type", either <ATARI> or <TDPDDS>, and the density, either "sd" or "dd". (See the I command for a discussion of the disk type.) This is an example of the summary display:

```
431/11=LISTED 276/53=FREE <TDPDDS>sd
```

As a special feature, TDP-DDS accepts the drive-number, alone, as a command to get a full Directory listing. For example:

```
  2<RETURN>
```

will give a Directory listing of all files on Drive-2.

SYNTACTIC ELEMENTS

filespec [D|X.X] - The specification of the files you want listed.

This parameter may be just a single file, a set of files using wildcards, or all files. It may also -- as a special case for this A command -- include a wildcard as the drive number to permit a search of all your drives. As an example of a wildcard drive number:

```
DX:X.TXT
```

specifies all .TXT files on all disk drives.

Note: the default filespec is: "ALL files on Drive-1".

/n (n=1:2:3:4:5:16) - The number of columns for the directory listing.

The /n option allows you to specify the number of columns in which the files will be displayed or printed. The standard 40-column screen can accommodate one or two columns. An 80-column board or printer can hold up to four columns. A 132-column printer can take up to six columns. At times you may prefer the 1-column format (/1) -- such as, when working with an alphabetized Directory.

The number of columns, once set, remains set until changed. As delivered, TDP-DDS is setup for 2 columns (/2).

You can change the default number of columns you get when you Boot-up TOP-DOS, from two columns to whatever you choose. First establish the desired number of columns by getting a Directory listing with that number of columns. Then use the H command to write out the DOS files to a disk. When you Boot-up from that disk, the A command will display a directory listing in the new number of columns.

NOTE: This option may not be used simultaneously with any other option. If you want to use the /n option to change the number of columns, and at same time to specify another option, you can achieve the same effect by a two-step process. First use the /n option to establish the number of columns you want, getting a sample listing. Then get the listing you want, using the other option.

HINT: To minimize Screen use, specify a non-existent file for the first listing.

/A - Alphabetize the File Directory on the Disk.

This option first alphabetizes the Directory on the disk and then gives the alphabetized listing. Thus, once alphabetized, the Directory remains so (until you add new files).

NOTE: this option works only on <TOPDOS> type disks.

This option may be used alone or with /C (either order is OK). If you want to use this option with another option (other than /C), get two Directory listings, first using the /A option and, second with the other option.

NOTE: This option uses program memory to sort the File Directory. (Only the upper C28 (hex) bytes are used.)

NOTE: The Directory is read into memory, sorted, and then written back on the disk. Thus, your disk must not be write-protected -- that is, the notch must not be covered. Otherwise, you will get an "ERR 144" (Not Ready) message.

/C - Compress the File Directory on the disk.

This option removes deleted files from the Directory, in order to speed Directory searches (which are required by all commands dealing with disk files). Use this option after you have deleted a large number of files from a disk in order to achieve a significant saving in search time.

The /C option is normally used with the /A option -- or it can be used alone. To use the /C option with another option (other than /A) use the technique described in the /A option.

NOTE: The notes under the /A option apply also to this option.

CAUTION: This option purges deleted files permanently from the Directory and prevents you from later Undeleting them.

/D - Find and List Deleted files.

This option lists deleted files in the File Directory on the specified disk drive. It is useful when you're not sure what happened to your files and may want to restore them.

The ability to be able to list deleted files in a disk's File Directory is a unique feature of TOP-DOS. This feature, of course, is essential to support the Undelete command, another unique TOP-DOS feature (see the U command).

This /D option may only be used alone.

NOTE: The /B and /D options list a special category of files, called "open" files, in addition to listing deleted files. Open files are files that were not properly closed for some reason, such as a power failure, or pressing the <SYSTEM-RESET> button at the wrong time. (In ATARI DOS 2, you cannot see these deleted and open files at all without using a disk utility program.)

Such open files are designated by an exclamation-point "!" before the filename on the Directory listing. The U command tells how to deal with such open files.

/B - Find and List Both "current" and deleted files.

This option will list both the "current" files and the deleted files.

NOTE: This option may only be used alone.

list-filespec [E:] - The filespec for the directory listing.

This parameter tells TOP-DOS where you want the directory listing to go. It can be "E:", the Screen Editor (which is the default), or "P:", the printer, or a disk file. If a disk file is specified, TOP-DOS checks to see if the file exists. If the file exists, you will be queried if it is OK to delete it. If you do not answer "Y<RETURN>", the command will be aborted. This feature protects against inadvertent deletion.

NOTE: Wildcards are inappropriate in the list-filespec.

/N - No query for list-file deletion.

This option suppresses the query for delete that would otherwise occur if a file by the same name as the list-file already exists. Use this option only when you are sure you want to override an existing file and do not wish to be queried.

EXAMPLES

If you type "ACRETURN" (giving no parameters), you will get the prompt:

```
DIR: filespec/n!A!B!C!D,listfile/N
```

If you then just type <RETURN> a second time, all current files on the default drive, Drive-I, will display on your screen, looking something like this:

```
DOS      .SYS 38  DUP      .SYS 64
MEDIT   .   79  MANUAL  .BAK 21
223/5=LISTED 484/59=FREE  <TOPDOS>
```

Note: The same result would be obtained by:

```
☛A DI:*.X<RETURN>
☛A DI:<RETURN>
☛A *.X<RETURN>
☛A <RETURN><RETURN>
```

which are all equivalent, due to the default conventions.

To select a certain group of files, use wildcards in the filespec. For example:

```
☛A *.SYS<RETURN>
```

will select only .SYS files to be listed.

If you specify a file that does not exist, you will get just the number of free sectors and free files, looking like this:

```
484/59=FREE  <TOPDOS>
```

The highlighted (inverse video) designation at the end of the display tells you which format your disk has: <TOPDOS> or <ATARI>. (See the I command for more information on this matter.)

A few more examples:

- ☛A /A/C<RETURN> Alphabetizes and compresses the Directory on Drive-I and lists it on the Screen
- ☛A *.TXT/A<RETURN> Alphabetizes the Directory on Drive-I, and lists the .TXT files on the Screen
- ☛A D2:,DirList<RETURN> Copies the directory listing from Drive-2 to the disk file DirList on Drive-1.
- ☛A ,D2:24Feb84<RETURN> Writes the directory listing from Drive-I to the file 24Feb84 on Drive-2.
- ☛A D3:,P:<RETURN> Prints the directory listing from Drive-3
- ☛A /B<RETURN> Lists Both "current" and deleted files
- ☛A /D<RETURN> Lists all deleted and open files

SYNTAX -- B

PURPOSE -- The B command is used to run the BASIC or other cartridge in your computer. Use it when you are done with your current TOP-DOS operations and want to go to the cartridge program.

DESCRIPTION -- If you have a cartridge in place, after you type "B <RETURN>", the prompt appropriate for that cartridge will appear on your screen:

READY for BASIC
EDIT for ASSEMBLER

If you have no cartridge inserted, a "No Cartridge" message will display.

SYNTACTIC ELEMENTS

The B command has no syntactic element, aside from the Command Letter.

EXAMPLES

▼B<RETURN>

transfers control to the cartridge.

SYNTAX -- C [source-filespec][,destination-filespec][/A:Q:N:M:U]

PURPOSE -- The Copy command is used to copy files from a source device to a destination device.

DESCRIPTION -- TOP-00S searches the File Directory for the source file you specified to be copied. Then TOP-DOS searches the destination device for a file of the same name and extension. If this file already exists, TOP-DOS will query you for permission to delete it before proceeding with the copy operation. (Without this check for existing destination files, files could be deleted without your knowledge.) This query may be suppressed by the /N option.

You also have the option (/Q) if you want to get a query before each file is copied (whether or not a destination file exists).

TOP-00S gives you a complete record of the copy operation by displaying a record on the screen for each file in a wildcard series, as it is processed. For example, the command:

▼C OLO_FILE.TXT,NEW_FILE.TXT<RETURN>

will produce the display:

Copy - 01:OLO_FILE.TXT

if a destination file of the same name does not exist, or:

Delete D1:NEW_FILE.TXT

if it does exist. If you chose to be queried about the copy, these display lines would be followed by a "?" and your "Y" or "N" response.

If a destination file exists and you do not use the /N option, TOP-00S will display:

Delete 01:NEW_FILE.TXT?

"Y<RETURN>" will cause the original NEW_FILE.TXT to be deleted and OLO_FILE.TXT to be copied as NEW_FILE.TXT. However, if you respond "N<RETURN>" or just <RETURN>, TOP-00S will skip the file and go on to the next file, if in a wildcard series or, if not, will return to the Command Prompt ▼.

If you try to copy a non-existent file, TOP-00S will simply skip the request and return to the Command Prompt ▼.

This command will copy files from one disk on one drive to another disk on a different drive, or to the same disk on the same drive.

When copying to a different drive, the names of the files may be changed in the process or they may be kept the same. However, when copying to the same drive, new names must be given to the copies, since they will reside on the same disk with the originals. (To copy a file from one disk to another, if you have only one drive, you must use the 0 Duplicate File command.)

Files may also be copied to the Printer "P:" or to or from the Screen "E:".

If you try to copy a file to itself, TOP-DOS will ignore the request.

To stop a copy command, the <BREAK>-Key can be used. The copy process will be interrupted at the next possible point, and the user will be queried if s/he wishes to abort the operation. A "Y" reply will cause the operation to terminate, and open files to be properly closed!

CONCATENATION

Another TOP-DOS feature is the ability to concatenate a wildcard series of files to the destination file, with a single command. This is accomplished by using wildcards in the source-filespec, but not in the destination-filespec. A new destination file will be created if a file of that name doesn't already exist. If one does exist, you will be queried for delete. A "Y<RETURN>" response will delete the old and create the new file, an "N<RETURN>" or just <RETURN> will simply append the source files to the destination file.

TOP-DOS is careful not to include the destination file as a source file, even if it matches the source-filespec.

SYNTACTIC ELEMENTS

source-filespec [D1:*.X] - The files you want copied.

destination-filespec [D1:*.X] - The filenames of the copies.

To copy a series of files to the same disk, the copies must have different names. For this case, wildcards in the destination-filespec must correspond to those in the source-filespec. For example, to make copies of your .BAS files to files with BAK extensions:

```
☛C *.BAS,*.BAK<RETURN>
```

will copy all the .BAS files to files of the same filenames, but with BAK extensions.

The same method can be used when copying a set of files to a different disk drive, For example:

```
☛C D1:*.BAS,D2:*.BAK<RETURN>
```

will copy all .BAS files from Drive-1 to Drive-2, but the copies will have the BAK extension.

When copying files from one drive to a different drive, you can maintain the same filenames and extensions on the destination disk. For example, either command:

```
☛C D1:*.BAS,D2:*.X<RETURN>  
☛C *.BAS,D2:<RETURN>
```

will copy .BAS files from Drive-1 to Drive-2. (The source device defaults to "D1:" and the destination filename and extension defaults to "X.X", since no filename or extension is specified.)

/A - APPEND to existing file.

This option appends the source file to an existing destination file. If the destination file does not exist, an "ERR 170" (File not found) message will be displayed.

Wildcards may be used in the source-filespec (but not in the destination-filespec) to append a series of files to another file. The files will be appended in the order that they appear in the Directory, as seen in an A command Directory Listing.

/Q - QUERY-for-copy of wildcard series.

This option causes TOP-DOS to query before copying each file, to give you the opportunity to select which files of a series you want copied.

/N - NO-QUERY-FOR-DELETE of destination file.

This option suppresses the query for delete that is ordinarily performed when a destination file exists and would be deleted by the copy operation. Use this option to speed up a copy operation, but only when you are sure you want to overwrite the destination files.

/M - MERGE files with the files on the destination disk.

This option allows you to copy only those source files that are not already on the destination disk. This makes an easy job of merging the files from one disk to another, without being interrupted by queries. If files from the source disk are also on the destination disk, they are skipped.

/U - UPDATE the destination files.

This option provides you with the ability to update a set of files on a destination disk from a set of files on a source disk. Only those files from the source disk that are already on the destination disk will be copied.

The /U option suppresses the query for delete, since the purpose of this option is to copy over the existing files on the destination disk. So the copy proceeds without interruption by queries.

EXAMPLES

```
▼C D2:,D1:<RETURN>
```

copies all files from Drive-2 to Drive-1.

```
▼C D1:ANYFILE.LST,E:<RETURN>
```

displays this file on the Screen.

```
▼C E:,ANYFILE.DEM<RETURN>
```

copies lines from the screen into a file named ANYFILE.DEM. Remember to type <RETURN> to enter each line and <CTRL>-3 (End of File) when all lines have been entered.

```
▼C ANYFILE.DEM,P:<RETURN>
```

prints this file.

A number of files can be concatenated to form a new file or appended to an existing file with one command line. For example:

```
▼C X.TXT,COLLECT.TXT<RETURN>  
▼C X.TXT,COLLECT.TXT/A<RETURN>
```

perform similar functions, except the first creates a new file, while the second starts with an existing file. The .TXT files will be appended in the order they appear in the Directory.

The commands:

```
▼C D2:,D1:/N<RETURN>  
▼C D2:,D1:/Q<RETURN>  
▼C D2:,D1:/M<RETURN>  
▼C D2:,D1:/U<RETURN>
```

copy files from Drive-2 to Drive-1. The first copies all files, without queries, deleting same-name destination files if they exist. The second queries on all files, giving you the chance to select the files you wish copied. The third merges the files from Drive-2 to the disk in Drive-1, copying only those files on Drive-2 which are not on Drive-1. The fourth updates the files on Drive-1 by copying only those files on Drive-2 which are already on Drive-1, deleting the old files on Drive-1, in the process.

SYNTAX -- D [filespec][/N]

PURPOSE -- The Delete Command disposes of disk files that you no longer need, so that their sectors and the space in the File Directory can be reused for other files. (To delete ALL files from a disk, it is much faster to use the I command with the V option.)

DESCRIPTION -- TOP-DOS searches the File Directory of the disk in the specified drive for each file you want to delete. For each file found, you are queried:

Delete D1:OLD_FILE.TXT?

(unless the /N option is given in the command). If you answer "Y <RETURN>", the file will be deleted. If you answer "N<RETURN>", or just <RETURN>, the file will not be deleted.

If you give the /N option, you will not be queried for delete. However, the above message will still appear to record the deletion, but the "?" will be absent and the deletion will be carried out without further action on your part.

SYNTACTIC ELEMENTS

filespec [D1:X.X] - The specification of the files you want deleted.

In addition to the usual use of wildcards in the filespec, a wildcard may be used as the drive number to allow you to delete a set of files on all your drives with one command.

/N - No Query for Delete.

This option will suppress the query for delete. TOP-DOS will delete each file as it is found in the search. Use this option to save time when you are sure you want to delete all files in a wildcard series and do not want to be queried individually about each deletion.

EXAMPLES

VD D1:X.BAK<RETURN>

deletes all files that have the BAK extension, asking you permission to delete each file as it is found in the Directory.

VD D1:X.BAK/N<RETURN>

performs the same operation as the previous example, except that the files are deleted without waiting for permission.

VD D:X:X.BAK/N<RETURN>

deletes all .BAK files from the disks in all drives, unconditionally.

SYNTAX -- E [D#:]old-filespec,new-filespec[/N:Q]

PURPOSE -- Rename allows you to change the names of files in a disk File Directory.

DESCRIPTION -- TOP-DOS searches the File Directory, changing the names of the files, as requested.

Before changing the name of each file, TOP-DOS checks to see if a file of the new name already exists. If it does, you are queried for permission to delete this file, before the rename process is performed. (This query for delete can be suppressed by the /N option.)

NOTE: Without this check for an existing file of the new name, and the deletion of this file before renaming the old file, you could end up with two files of the same name, as is possible in the ATARI DOS.

You also have the option (/Q) to get a query before each file is renamed, whether or not a file of the new name already exists.

TOP-DOS gives you a complete record of the rename operation by displaying a record on the screen for each file it processes in a wildcard series. For example:

♥E OLD_FILE.TXT,NEW_FILE.TXT<RETURN>

will give the display:

♥Rename D1:OLD_FILE.TXT

if a file named NEW_NAME.TXT does not exist, or:

♥Delete D1:NEW_NAME.TXT

if it does exist. If you chose to be queried about the rename, these display lines would be followed by a "?" and your "Y" or "N" response.

If you try to rename a non-existent file, TOP-DOS will simply skip the request. If more files remain in a wildcard search, TOP-DOS will continue with the next file. Otherwise the command is finished and you will get the ♥ prompt for the next command.

SYNTACTIC ELEMENTS

D#: [D1:] - The device designator of the disk drive to be used.

For this E command, only, the disk-drive number specification applies to both the old and new files. Since in this command, only the entries in the File Directory are changed, no files are moved. Therefore, only one disk is involved in the operation.

old-filespec - The filespec of the files you want to rename.

The above disk-drive specification applies to this old-filespec.

new-filespec - The new names of the files you want renamed.

This element has the format of a filespec, but without the disk drive specification. The above disk-drive specification applies to this new-filespec, as well as the old-filespec, as explained above.

/Q - QUERY for rename of each file of a wildcard series.

This option allows you to choose which files of a wildcard series will be renamed. You will be queried, as TOP-DOS finds each file in its search of the File Directory. If a file of the new-name does not exist, the query will appear:

Rename D1:OLDFILE.TXT?

"Y<RETURN>" will cause the file to be renamed. "N<RETURN>" or just <RETURN> will skip this file.

If, however, a file already exists of the new-name, the query will appear:

Delete OLD_FILE.TXT?

If you reply "Y<RETURN>", that file will be deleted and the old-name file will be renamed to the new-name. A "N<RETURN>" or just <RETURN> will cause the file to be skipped.

/N - NO QUERY FOR DELETE.

This option suppresses the query for delete and speeds up the rename operation. Use this only if you know ahead of time that you want all files to be renamed regardless of whether or not existing same-name files will be deleted.

EXAMPLES

☛ D2:OLD_FILE.TXT,NEW_NAME.TXT<RETURN>

changes a file on Drive-2 from old to new name. If a file by the name of NEW_NAME.TXT already exists on Drive-2, you will be queried for permission to delete this existing file before OLD_FILE.TXT is renamed to NEW_NAME.TXT.

☛ D2:OLD_FILE.TXT,NEW_NAME.TXT/N<RETURN>

Performs the same operation as the above example, except the rename will be done without query. If the file NEW_NAME.TXT already exists, it will be deleted.

☛ X.TXT,X.ASC/Q<RETURN>

renames all .TXT files on Drive-1 to .ASC files, with a query before each rename.

SYNTAX -- F filespec

PURPOSE -- The Lock command is used to protect them from being deleted or changed.

DESCRIPTION -- The files specified by the filespec are marked "locked" in the File Directory. Any subsequent operation (except the I and J commands) which would delete or modify them will be aborted and an "ERR 167" (File locked) message will be issued.

Locked files are indicated in a Directory Listing with a "+" in front of the file name.

Locked files can be unlocked with the G Unlock command.

CAUTION: The I Initialize Disk command and the J Duplicate Disk command effectively wipe out all files on the destination disk, by virtue of their mission. They do not check for locked files.

SYNTACTIC ELEMENTS

filespec - The specification of the files you want locked.

Note that there is no default for this parameter. The absence of a default is a feature which prevents your locking all the files on Drive-1 by accident. This could easily occur, if you hit the "F" key by mistake, and then press <RETURN> twice to get out of the command. The second <RETURN> would trigger the default filespec, if this command had such a default. Since the standard default filespec is [D1:*.X], all files on Drive-1 would be locked.

Although you must give a filespec, you may leave out the drive designator, to get the default Drive-1. For example:

```
♥F any_file.TXT<RETURN>
```

Or you may just give the drive designator and leave out the filename and extension to get the default "*.X". For example:

```
♥F D2:<RETURN>
```

EXAMPLES

```
♥F D1:ANYFILE.TXT<RETURN>
```

locks this one file.

```
♥F X.BAS<RETURN>
```

locks all files with a BAS extension on Drive-1.

```
♥F D2:<RETURN>
```

locks all files on Drive-2.

SYNTAX -- G filespec

PURPOSE -- This command unlocks a previously locked file or set of files, allowing deletion, renaming, appending, or editing to take place.

DESCRIPTION -- When a file is unlocked, the "+" will vanish from the Directory listing.

SYNTACTIC ELEMENTS

filespec - The specification of the files you want unlocked.

As in F command, there is no default for this parameter. This is a feature which prevents your unlocking all the files on Drive-1 by accident. This could easily occur, if you hit the "G" key by mistake, and then press <RETURN> twice to get out of the command. The second <RETURN> would get the standard default filespec [D1:X.X] if it applied to this command.

Although you must give a filespec, you may leave out the drive designator, to get the default Drive-1. For example:

♥G any_file.TXT<RETURN>

Or you may just give the drive designator and leave out the filename and extension to get the default "X.X". For example:

♥G D2:<RETURN>

EXAMPLES

♥G D1:ANYFILE.TXT<RETURN>

unlocks this one file.

♥G X.BAS<RETURN>

unlocks all files with a BAS extension on Drive-1.

♥G D2:<RETURN>

unlocks all files on Drive-2.

SYNTAX -- H D#

PURPOSE -- The Write OOS Files command is used to write the two TOP-OOS files: OOS.SYS and OUP.SYS onto a disk. This enables you to Boot-up TOP-DOS from any disk, and also makes it possible to incorporate any customizing features you may wish to make permanent features of your TOP-DOS operations. (See the S command for a list of these customizing choices.)

DESCRIPTION -- This command writes TOP-OOS from memory to OOS.SYS and OUP.SYS, linking OOS.SYS to the Boot sectors (see Appendix A).

If you do not specify a drive number, D#, TOP-OOS will prompt you to give one. When you give a drive number, TOP-DOS queries you to be sure you want to complete the operation, since it will overwrite any OOS you may have on your disk. A "Y" reply is required to proceed.

The message "Writing OOS" will let you know it's working. When the Command Prompt * again appears you will know the operation is finished and you may continue with your next command.

NOTE: If you need the RS232 handler for telecommunications, then the file RS232.TOP, or its renamed version RS232.AUT, must be separately copied to your disk using the C or O command. Similarly, if you want to use the HELP facility from your new disk, you should also copy the .### files, using the C or O command. If you decide not to include all these HELP files, you might want to include at least the general HELP file "#.###".

The H command writes TOP-OOS as it currently exists in memory at the time this command is executed. Be sure you have the various features and options set the way you want them to be at Boot-up (see the N and S commands). In addition, refer to the NOTE in the description of the /M option of the L command to see how to control the display of a memory map when the OUP.SYS and .AUT files are loaded.

SYNTACTIC ELEMENTS

D# - The disk drive number (1-8).

The disk drive number must be given. There is no default. (Although it is not required by this command, you may type the letter "D" before the drive number. Example: "D1" or "1".)

EXAMPLES

*H 1<RETURN>

will write the TOP-DOS files to the disk in Drive-1 -- assuming that a "Y" reply is given when permission is requested to proceed.

SYNTAX -- I D[*VDI*],A[*n*/8]

PURPOSE -- The I command is used to format a new disk and to initialize it with a VTOC and a File Directory. It can also be used to re-initialize a previously-used disk that no longer contains useful files.

DESCRIPTION -- The I command physically formats a new disk so that data can be stored on it. After a disk is formatted, it is initialized to enable it to accept files, by creating 9 special sectors: 1 sector for the VTOC (Volume Table of Contents) and 8 sectors for the File Directory.

To restore an already formatted disk and to save the formatting time, the I command may be used, with the V option, to perform only the initialization process.

A disk may be initialized in either the <TOPDOS> or <ATARI> "type" (see Chapter Four).

If the disk type is <ATARI>, then the file structure includes file numbers in the sectors of a file. If the disk type is <TOPDOS>, then file numbers are not included. (See the description of the A option below for more information.)

Since the I command will effectively erase all files from a disk, TOP-DOS queries you for permission, before proceeding.

NOTE: The disk will be formatted in the density for which the drive is set. Use the S command to both check and set the density of your drive before using the I command.

If there are bad sectors on a disk, "ERR 173" will display, and TOP-DOS will not format the disk.

SYNTACTIC ELEMENTS

D# - Disk drive number (1-8).

V [format] - Initialize the VTOC only.

This option skips the physical disk formatting process and performs only the initialization of the VTOC and the File Directory on the disk. (Once a disk has been formatted, it does not have to be reformatted.) This cuts the time from 48 to 3 seconds. Use this option when to initialize a disk that has been previously formatted and want to save time.

If V is not given, the command defaults to the performance of a full format operation on the disk.

NOTE: This option is NOT preceded by a Slash "/" character (as are options on filespecs).

CAUTION: Do not use the V option if you want to initialize a disk in a different density than it was previously formatted in. In this case, you must do the full format.

A [⟨TOPDOS⟩-type] - Initialize the disk as ⟨ATARI⟩-type.

This parameter is used if you want the initialization process to produce an ⟨ATARI⟩-type disk. If the parameter is not given, the default, ⟨TOPDOS⟩, type is assumed.

An ⟨ATARI⟩-type disk is needed if you want to be able to read its files with the ATARI DOS. Otherwise, use the default ⟨TOPDOS⟩-type, which provides more flexibility.

A ⟨TOPDOS⟩-type disk allows TOP-DOS to alphabetize and compress the File Directory, to handle double-sided double-density drives, and to access up to 1968 sectors on 8" double-density drives.

TOP-DOS can read and write both ⟨ATARI⟩- and ⟨TOPDOS⟩-type disks. It can duplicate them, and can copy files between them.

The disk type is displayed in inverse video at the bottom of a Directory Listing.

Use the A parameter only if it is important to you to be able to read a disk with Atari DOS, which cannot read a ⟨TOPDOS⟩ type disk. This "backwards compatibility", for example, allows you to send a disk to a friend who doesn't have TOP-DOS yet.

NOTE: The A parameter must be preceded by a Comma "," but not by a Slash "/".

n/8 - Set the number of sectors on the disk.

This is a special parameter for the ADVANCED USER.

If you have a nonstandard drive, this parameter will enable you to control the number of sectors in your UTOC.

"n/8" is the number of sectors you require divided by 8 (ignore any remainder). The maximum number of sectors is 944 for single-density and 1968 for double-density disks. The corresponding values for this parameter are $944/8=118$ and $1968/8=246$, respectively.

By way of comparison, a single-sided drive has 720 sectors. The corresponding value for this parameter is $720/8=90$.

NOTE: This parameter must be preceded by a Comma ",",

EXAMPLES

For a previously formatted disk, if you want to change the density, type:

❖I IV<RETURN>

to initialize the disk on Drive-1 as <TOPDOS> type, without invoking the physical formatting process.

❖I 2<RETURN>

formats and initializes the disk on Drive-2 as <TOPDOS> type.

❖I 2,A<RETURN>

same as above, except as <ATARI> type.

❖I 2V,A<RETURN>

same as above, except the disk formatting operation is suppressed and only the UTOC initialization is performed.

❖I 2,246<RETURN>

formats and initializes the UTOC on the disk on Drive-2 as <TOPDOS> type, with $1968=246 \times 8$ sectors.

❖I 2V,246<RETURN>

same as above, except only the UTOC initialization is performed.

SYNTAX -- J source-drive-#,destination-drive-#

PURPOSE -- The Duplicate Disk command is used to create an exact duplicate of a whole disk, with the same files, sector numbers, VTOC, and File Directory.

DESCRIPTION -- TOP-DOS copies only sectors which are in use, as marked in the VTOC. This process is faster than copying all files on the disk, and is especially useful if you have only one disk drive.

After typing in the complete command, TOP-DOS issues clear prompts to guide you through the process. If you use two disk drives, the process is simple and fast. The prompt tells you to insert both disks and type <RETURN>.

If you have only one drive, you must alternate between the source disk and the destination disk, swapping disks as directed by the prompts.³⁶ The amount of time it takes for the duplication operation depends only on the size of your files.

NOTE: TOP-DOS will detect if your destination disk is not formatted in the same density as your source disk and will automatically format it for you. However, this detection causes a delay and possibly a timeout, so you should normally use a correctly formatted disk.

If you have trouble at the end of duplicating a double-sided double-density disk that was formatted on a PERCOM drive, your drive may have an old ROM. You can circumvent the problem by using 179 as the second parameter of the I command when you initialize the disk. For example, use "I 1,179<RETURN>" when initializing double-sided disks on that drive.

CAUTION: The J command uses program memory as a buffer.

SYNTACTIC ELEMENTS

source-drive-# - The disk drive number of the source.

This parameter is the number of the disk drive that you will use for the source disk to be duplicated. This parameter must be given; there is no default.

destination-drive-# - The disk drive number of the destination.

This parameter is the number of the drive that you will use for the duplicate disk. It may be the same as the source-drive-#. This parameter must be given; there is no default.

EXAMPLES

♥J 1,2<RETURN>

will duplicate the disk on Drive-1 to the disk on Drive-2.

♥J 1,1<RETURN>

will duplicate a disk by requiring your swapping the source disk and the destination disk several times.

SYNTAX -- K filespec[/AIN][,start,end][,init][,run]

PURPOSE -- This command is used to save the contents of an area of memory in binary format.

CAUTION: This command is for ADVANCED USERS who are familiar with hex notation.

NOTE: All memory addresses are in hex.

DESCRIPTION -- TOP-DOS writes a binary file from the area of memory specified by the parameters of the command. The binary file can later be read back to the same area of memory, using the L Load command.

Through the use of the /A option, a memory area can be appended to an existing file. In this way, you can build up a file with a number of "sections", each of which contains the contents of an area of memory.

Provision is also made for either or both a "Run" address and an "Initialization" address, which allow a binary file to be loaded and automatically executed, either as a whole program, or in sections as they are read in to memory.

As a safety measure, TOP-DOS checks to see if the binary file to be written already exists -- unless the /A Append option or the /N No-query-for-delete option is specified. If the file exists, and neither option was given, you will be queried for permission to delete the file. Without this check, you could lose a file unintentionally.

SYNTACTIC ELEMENTS

filespec - The specification of the binary file.

This parameter is the filespec of the binary file to be written from the binary data in the memory area. At least the filename and extension must be given. However, the disk drive specification can be omitted, in which case it will default to "D1:".

/A - Append to an existing file.

This option specifies that the binary data be appended to an existing file, as a separate section. If the file does not exist, an "ERR 170" (File not found) will be displayed.

/N - No-query-for-delete of file.

This option eliminates the query which protects you from the accidental deletion of an existing file. Use it only if you are sure you want to overwrite a file, if it already exists.

start - The first address of the memory area.

This parameter gives the first location, in hex, of the memory area to be written to the binary file. It may be omitted, only if the "end" address is also omitted, in which case no area of memory will be written. If these addresses are omitted, their two commas " ," must be included to indicate the absence of these address parameters.

end - The last address of the memory area.

This parameter gives the last location, in hex, of the memory area to be written. Refer to "start" above for comments on omitting this parameter.

init - The initialization address.

This parameter gives the location, in hex, in memory where the section just written will be executed immediately after being loaded by the L Load command. If this parameter is omitted, no initialization address will be written to the file. If the init address is omitted, but the run address is given, then the comma " ," before the init address must be given.

Although not often used, this parameter can be extremely valuable since it allows a multi-section binary file to be loaded and executed in pieces.

run - The run address.

This parameter gives the hex location in memory where the whole program will be executed after an L command load is complete. If this parameter is omitted, no run address will be written to the file.

NOTE: Although all of the above parameters, except the filespec, may be individually omitted, at least one must be given. If not, the command has no function and TOP-DOS will not execute it and just return to the command prompt ♣.

EXAMPLES

♣K D2:ANYFILE.SAV/A,5000,5FFF,,5000<RETURN>

will append the memory area from 5000 through 5FFF hex to the file ANYFILE.SAV on Drive-2 and add a run address at 5000 hex.

♣K OLD_FILE.SAV/A,,,5000<RETURN>

will add just a run address at 5000 hex to the file OLD_FILE.SAV.

♣K NEW_FILE.SAV/N,6000,6FFF<RETURN>

will create the file NEW_FILE.SAV on Drive-1, deleting a file of that name if it exists, and write memory from 6000 through 6FFF hex.

SYNTAX -- L filespec[/N/OIM]

PURPOSE -- The Binary Load command is used to load and run binary files.

DESCRIPTION -- TOP-DOS searches the File Directory on the designated disk for the specified load file. It then loads the file, section by section.

Each section is loaded at the memory address which is included on the disk at the beginning of that section. After each section is loaded, TOP-DOS checks to see if an "Initialization" address is present and if it is, control is transferred to that address.

At the end of the load, control is transferred to a "Run" address, if one was included in the file. (If more than one Run address is in the file, the last one is used.)

The "Initialization" and "Run" addresses are discussed in the K Binary Save command, and that command is one source of binary files. Other sources are the ATARI Assembler Cartridge, other assemblers, compilers, and assembly-language programs purchased on disk.

The execution of the Initialization and/or Run addresses may be suppressed by the /N and /O options, as described below.

A memory map of the areas loaded may be obtained through the use of the /M option.

NOTE: If you encounter a binary file produced by OS/A+ (version 2.1) or DOS-XL, you will find that this L command will load, but not run it. To run this file, load it, using the /M Map option. Note the first memory location. Then use the M command to run it at that address. Or you can use the K command to append the run address to your file, so it will run automatically when loaded.

SYNTACTIC ELEMENTS

filespec - The specification of the binary file to be loaded.

This parameter gives the disk file to be loaded. It does not have a default, although the disk drive-# may be omitted to get the "Dii" default drive.

If the file is not a binary file, the message "Not Load File" will be displayed.

/N - No initialization or run.

This option suppresses the execution of the Run address and any and all Initialization addresses. Use this option if you want to load a file but not run it. It can later be run using the M command.

/O - Only Initialization (no run).

This option suppresses the execution of the Run address at the end of the load, but permits the execution of any and all Initialization addresses. Use this option when you want to load a file but not run it, and where the initialization is necessary for the loading process to proceed.

The program can later be run, using the M command.

/M Memory Map.

This option gives a depiction of what areas of memory are loaded. As each section of the binary file is loaded into memory, the first and last (hex) addresses of that memory area are displayed. If the file includes several sections, the memory area of each section is so displayed.

This option can be used with either the N or O options.

NOTE: When TOP-DOS loads the DUP.SYS or an .AUT file, whether a memory map is displayed depends on whether you used the /M option with the last use of the L command. If you want to see the memory map when these files are loaded, then load a file with the /M option (a non-existent file will do) before writing DOS files with the H command. Conversely, if you do not want the memory map on when the DUP.SYS and .AUT files are loaded, be sure that the last L command load, before writing DOS files, is done without the /M option.

CAUTION: A Binary Load will cause BASIC to reinitialize if reentered after the Load operation. This effect is designed to protect against the corruption of a BASIC program by the loaded file. If you have a BASIC program in its buffer, you should SAVE it before entering the TOP-DOS Menu command program. However, if you want to use a binary file in conjunction with a BASIC program, you should load the binary file first, change MEMLO and/or MEMTOP to keep BASIC from using the area of your binary program, and then call BASIC with the B command.

EXAMPLES

♥L TOOL.SAV/M/N<RETURN>

loads the binary file TOOL.SAV from the default Drive-1, with a memory map, and with execution suppressed.

♥L D2:TOOL<RETURN>

loads the file TOOL from Drive-2 and executes the program, if the file includes a Run address.

♥L TOOL/O<RETURN>

shows the use of the /O option, with section Initialization only. The Run feature is suppressed.

SYNTAX -- M [hex-address]

PURPOSE -- The M command is used to run a program already in memory.

DESCRIPTION -- The M command starts the execution of a program in memory at the address given as parameter to the command. If an address is not given, it defaults to the Run address of the last binary file loaded, or to the address of the previous call of the M command, whichever was more recent.

The default address will display on your screen as part of the Parameter Prompt. If no default is available, <NONE> will display.

CAUTION: After <SYSTEM-RESET>, the default address is set to 0. The use of this 0 default address will produce unpredictable results and should be avoided.

SYNTACTIC ELEMENTS

hex-address [default-address] - Memory address to start execution.

This parameter is the location in memory, given in hex notation, where TOP-DOS will transfer control to, at the time you type <RETURN>.

The default-address is the Run address of the last binary file loaded, or the address of the previous call of the M command, whichever was more recent.

EXAMPLES

♥M 5000<RETURN>

transfers control to location 5000 hex in memory to begin the execution of the program which had been previously loaded or stored there.

♥M<RETURN><RETURN>

transfers control to the Run address of the last binary program loaded or to the address of the last call of the M command, whichever was more recent. Note that two <RETURN>s are required to get the default address. The first <RETURN> results in the prompt:

RUN: addr [Run Addr=5000]

if the first example above had been previously executed. The second <RETURN> causes the program at 5000 hex to be executed.

If a program had not been previously loaded and this is the first M command used, the prompt would appear:

RUN: addr [Run Addr=<NONE>]

indicating that there is no default run address. If you hit another <RETURN> at this point without specifying an address, TOP-DOS will ignore the command and simply return to the Command Prompt ♥.

SYNTAX -- N F|O|C

PURPOSE -- The N command is used to control the "MEM.SAV" facility which allows user programs to share the area of memory used by the TOP-DOS Menu command program, which is loaded from the DUP.SYS file.

DESCRIPTION -- The MEM.SAV facility, which is controlled by this command, is fully described in Appendix B.

SYNTACTIC ELEMENTS

F = OFF - turns the MEM.SAV facility OFF.

This parameter disables the MEM.SAV facility. Turn it OFF to save time when you don't need it.

O = ON - turns the MEM.SAV facility ON.

This parameter activates the MEM.SAV facility. Turn it ON when you want to be able to go from BASIC (or another user program) to the TOP-DOS Menu-command program and back again to BASIC, without losing your BASIC program.

C = CLEAR - "clears" the MEM.SAV facility.

This parameter is intended for the ADVANCED USER. It signals TOP-DOS that the MEM.SAV file does not hold valid data. Use this parameter when you know that the data in the MEM.SAV file is obsolete, but want to leave the facility ON. This will save the time it takes to load the MEM.SAV data back into memory.

EXAMPLES

ON O<RETURN>

turns the MEM.SAV facility ON.

ON C<RETURN>

signals TOP-DOS that the data in the MEM.SAV file is obsolete.

SYNTAX -- 0 [filespec][/*Q*][,sd]

PURPOSE -- This command is only used if you have just one drive and want to copy files from one disk to another.

DESCRIPTION -- After you enter the 0 command, TOP-DOS prompts you to insert your source disk and type <RETURN>. TOP-DOS then searches the File Directory for the first file specified. If you give the /*Q* option you will be queried for permission to copy this file.

Each file is copied individually by reading the file from the source disk into computer memory. Then you are instructed to insert the destination disk and type <RETURN>. The data from memory is then written to the destination disk as a file of the same name.

If the file is too large to fit in program memory, you will be instructed to again insert the source disk, and later, the destination disk, until the whole file is transferred.

CAUTION: All of program memory is used as a large buffer for the copy operation.

The protection against inadvertent file deletion provided by the C Copy command, is not available for the "0" command. It is not feasible for the "0" command to check if a file of the same name as the source file already exists on the destination disk. Therefore, if such a file does exist, it will be deleted in the copy process and will be replaced by a copy of the file from the source disk.

SYNTACTIC ELEMENTS

filespec [D1:X.X] - The specification of the files to be copied.

This parameter specifies which file or files you want copied from one disk to another.

/Q - QUERY for copy of wildcard series.

This option causes TOP-DOS to query before copying each file to give you the opportunity to select which files of a series you want copied.

To accept a file to be copied, type "Y<RETURN>". To reject a file, type "N<RETURN>" or just <RETURN>.

sd - Source and Destination Densities.

This parameter consists of two of the letters: S, D, or Q (Single-density, Double-density, or Quad-density). (See the S command for information on setting your drive to these densities.)

The first letter represents the source density and the second, the destination density. Any combination is permissible, including two of the same letters.

If this parameter is given, the drive density will be switched automatically before each disk is to be inserted.

At the end of the O command operation, if wildcards were used, the drive will contain the source disk and the drive will be set to the source density. If wildcards were not used, the drive will contain the destination disk and the drive will be set to the destination density.

EXAMPLES

▼O OLD_FILE.TXT<RETURN>

will duplicate (copy) the file from a "source" disk to be placed in Drive-1 to a "destination" disk to be later placed in the same drive.

▼O *.TXT<RETURN>/Q

will copy all *.TXT files from a source disk to a destination disk, both to be placed in Drive-1, alternately. Before each file is copied, you will be queried as to whether you want to copy that file.

▼O OLD_FILE.TXT,SQ<RETURN>

copies the file from a single-density disk to a "quad"-density (double-sided double density) disk.

SYNTAX -- P [hex-address]

PURPOSE -- This command is used to initiate the running of a program already in memory. It differs from the M command, which is similar, in how the default is handled. The P command is most useful in executing a frequently used program, quickly.

DESCRIPTION -- The P command starts the execution of a program in memory at the address given as parameter to the command. If the address is not given, it defaults to the address given in the previous use of the P command.

The P command differs from M in two ways:

1. The P command does not default to the Run address of a load file, as does M.
2. The P command does not give a parameter prompt, if only the Command Letter is given, as does M, but rather begins execution of the memory program immediately.

So to execute a frequently-used program, simply call it once by typing "P hex-address<RETURN>". Then for subsequent calls, merely type "P<RETURN>".

You can make a default address permanent, by the writing TOP-DOS to a disk with the H command. The next time you boot-up TOP-DOS from that disk, the default start address for the P command will already be established.

SYNTACTIC ELEMENTS

hex-address [default-address] - Memory address to start execution.

This parameter is the location in memory, given in hex notation, where TOP-DOS will transfer control to, at the time you type <RETURN>.

The default-address is the last address given in a previous use of the command. If an address had never been given, then TOP-DOS simply returns a Command Prompt ♡.

EXAMPLES

♡P 5000<RETURN>

initiates the execution of a program already loaded in memory at hex location 5000. Subsequent calls:

♡P<RETURN>

will cause the same action.

SYNTAX -- Q filespec

PURPOSE -- This command file capability allows you to execute a complex sequence of TOP-DOS Menu commands in one simple operation. It is particularly useful in executing repetitive sequences of commands that you use often.

DESCRIPTION -- The Q command reads the specified command file, one line at a time. Each line is displayed on the screen and interpreted just as though you had entered it from your keyboard.

How to Create a Command File

A command file is a text file, made up of lines, just as you would type them into TOP-DOS. It can be created by any text editor, or it can be created on the screen and copied to a disk file using the C command.

How to Include Comments in a Command File

In addition to commands, you can include comments or other messages, which will be displayed on the screen as the command file is executed. Comment lines start with a space (so they will not be interpreted as commands).

A Command File can Include Q Commands!

After you use command files for a while and discover their power and utility, you may find that you want to create a command file that includes calls to other command files. TOP-DOS permits this "recursion" to two levels. That is, you may create one command file that calls a second-level command file.

How to Handle Commands that Require Responses to Queries

Sometimes there is the need in a command file for answers to questions that TOP-DOS displays on the screen, as it executes your commands. For example, if you call the I command to initialize a disk, TOP-DOS will query for permission to proceed and expect a "Y" or "N" line in response (see the I command).

You have a choice as to how to handle the response. You can set up your command file so that IT will supply the response. Or you can set it up so that YOU will supply the response -- that is, TOP-DOS will wait for you to give the response from the keyboard. The choice is made by including one of these special control characters: "\$" or "#" in your command file.

"\$" and "#" Control Characters

"\$" and "#" are special commands used to control the source of "secondary" input, when needed by Menu commands executed within a command file.

By "secondary input" is meant any input required by a Menu command, which is not given on the command line. This would include parameters, if not given on the command line, and responses to a queries, such as required by the H and I commands.

When used, one of these special commands (\$ or #) must appear in the first column of a line. The rest of the line is not used by TOP-DOS, so you may use it for comments if you wish.

For example:

```
❖I 2<RETURN>
```

to format disk drive-2 requires a subsequent "Y" confirmation to enable the operation.

Whether this secondary input is taken from the next line of the command file, or from the user via the Screen, is determined by which special command "\$" or "#" was LAST used. (You can switch between them throughout a command file.)

The special command "\$" causes TOP-DOS to seek secondary input from YOU (via the Screen Editor "E:"). Whereas the special command "#" causes TOP-DOS to seek secondary input from the command file itself.

NOTE: To help remember which is which: People like dollars (\$), but computers like numbers (#). So "\$" gets secondary input from the user and "#", from the command file.

How to Interrupt a Command File

If you want your command file to stop when it gets to a certain line, use the "!" character at the beginning of the line.

When "!" is the first character on a line in a command file, TOP-DOS will interrupt the process and will wait until the user types <RETURN>. This gives the user a chance to read and respond to the message on the screen before proceeding to the next line. TOP-DOS ignores the rest of the "!" line, so you can use it for a message.

SYNTACTIC ELEMENTS

filespec - The command file containing TOP-DOS commands & comments.

EXAMPLES

Suppose you want to create a command file to:

- o Alphabetize and compress all directories,
- o Initialize a new disk on Drive-2,
- o Write TOP-DOS to the newly-initialized disk on Drive-2,
- o Get a listing of all .TXT files on Drive-1,
- o Copy all .TXT files from Drive-1 to Drive-2,
- o Lock all .TXT files on Drive-2,
- o Unlock all .TXT files on Drive-1,
- o Delete all .TXT files on Drive-1,
- o Get a new Directory Listing of the .TXT files in the Directories on all disk drives, and
- o Finally, get a complete printout of all .TXT files on Drive-2.

Since the command file for this example is short, you can easily create it on the screen. To use this method you would first enter the command:

```
♥C E:;UPDATE.CMD<RETURN>
```

to copy your input from the Screen to the file UPDATE.CMD: After a few seconds, when the disk activity settled down, you would enter the following lines:

```
    This command file initializes a disk<RETURN>
    on D2, copies .TXT files from D1,<RETURN>
    deletes the old files from D1, and<RETURN>
    locks & prints the new files on D2<RETURN>
A DX:/A/C<RETURN>
! Insert a blank disk on Drive-2 and Type <RETURN>
$ You will supply the response to queries hereafter<RETURN>
I 2<RETURN>
# The response to queries hereafter comes from this file<RETURN>
H 2<RETURN>
Y<RETURN>
C D1:*.TXT,D2:<RETURN>
F D2:*.TXT<RETURN>
G D1:*.TXT<RETURN>
D D1:*.TXT/N<RETURN>
A DX:<RETURN>
C D2:*.TXT,P:<RETURN>
<CTRL>3
```

The first few lines are comments which tell what this command does. The comment lines in this example have four spaces at the beginning, even though only one is required. The command lines are flush with the left margin on the Screen, as they should be.

As each line is typed on the Screen, and edited if necessary, a <RETURN> enters it into the command file. After all lines are entered, <CTRL>-3 (End-of-File) signals TOP-DOS to close the file.

In this example, the special command "\$" is used before the I (Initialize disk) command to return control to the user, so that s/he can be sure that a disk with valuable files is not destroyed. Once the disk is initialized, the H (Write DOS Files) command can proceed without the need for further user intervention. Therefore, the special command "#" is used before the H command to return control to the command file.

The next time you wish to perform these operations, you need only type the single line:

```
♥Q UPDATE.CMD<RETURN>
```

to execute the entire sequence of commands in this UPDATE.CMD file.

SYNTAX -- R hex-address

PURPOSE -- The Read/Store Command gives you a convenient way to examine and replace bytes in memory, without using BASIC's PEEK, POKE, and PRINT commands or resorting to an Assembler or a utility program.

DESCRIPTION -- The R command is called by giving a hex memory address. TOP-DOS responds by displaying the contents of the 8 bytes of data at that location in memory, in both hex and ATASCII (character) form.

NOTE: All addresses and byte values are given in hexadecimal notation.

For example, if you type:

•R 5040<RETURN>

the display might look something like this:

5040 52 45 41 44 44 2F 53 54 52 READ/STR
X=Store █

The 5040 is the memory address. The two-digit hex numbers are the contents of the eight bytes of memory, starting at 5040. The "READ/STR" are the eight character representations of the eight bytes.

The prompt "X=Store" (read "Star=Store") reminds you how to arm the store mode. You can either:

- o Type <RETURN> which causes the next 8 bytes to display. This may be repeated as often as you wish, thus requiring only a single keystroke to advance through memory.
- o Type "-<RETURN>" to move backwards in memory, to display the previous 8 bytes. Once you have backed up in memory, subsequent <RETURN>s will continue to go backwards in memory. To go forward again, type "+" before the next <RETURN>
- o Type "/"<RETURN>" or <BREAK> to EXIT the R command.
- o Type the next TOP-DOS command you want executed. This is a quick way of exiting the R command, since the R command exit and the next command are accomplished at the same time and on the same line.
- o Type "X<RETURN>" to arm the store mode to change bytes in memory.

Store-Mode to Change Bytes in Memory

TOP-DOS responds to the "*" by blinking the cursor to warn you that the store mode is armed. (If there are any inverse video characters on the screen they will also blink.) What you do now may change memory and could cause a system "crash" if you happen to change a byte used by TOP-DOS or the ATARI OS. That is why TOP-DOS requires you to arm the store mode and also why the cursor blinks to alert you that the store mode is armed.

NOTE: A "crash" doesn't mean your computer will be damaged, just that the system will "hang up" so that you can not continue what you are doing. You may be able to recover with a <SYSTEM-RESET>. If not, you will have to re-boot, destroying any data or program you may have in memory. So, be careful when the cursor blinks!

With the store mode armed, you can use the Screen Editor cursor commands and other facilities to edit bytes on the screen. Just move the cursor to the bytes you want to change and type the new values over them. Edit one line at a time and then type <RETURN> to change the memory bytes on that line to the new values.

TOP-DOS will then display three or four lines:

- o The original values of the bytes before the change.
- o The line as you edited it.
- o An error message if you entered an invalid hex number.
- o The new values as now stored in memory.

These lines may appear confusing at first. However, they provide you with a complete picture of the process.

If all went as expected, all you need is the last line. The other lines are displayed to help you, if something did not go as expected.

In the first line, you can see the byte values before the change. In the second line, you can see what the line looked like when you hit <RETURN>.

Furthermore, you can, if need be, restore the original values of the bytes without retyping them. Just position the cursor to the first line and hit <RETURN>. The original values will be restored!

To leave the restore mode -- and stop the cursor from blinking -- move the cursor down to a blank line, preferably at the bottom of the screen, and hit <RETURN>. (Delete a line, if you have to.) You can also abort out of the R command completely by pressing the <BREAK> Key.

HINT: You may find it helpful to "clean up" the display after editing and changing a line of bytes, by deleting lines you no longer need.

If you like the bytes as they are now in memory, as shown in the last line, move the cursor up to the first line and delete the first two lines (with <SHIFT>-<DELETE>), one at a time. This will leave only the line containing the current values of the bytes.

If you do not like the bytes in the last line, then choose one of the three lines to start with. Edit that line and hit <RETURN>. This line will become three or four lines, leaving you with a total of five to seven lines from the original line, back two changes ago. Clean up the display by deleting lines.

If the display gets too confusing, it might be best to exit the R command, as described above. Then call the R command again to display the bytes in question. You should have enough history on the screen to see both the old display, as well as the new. If you want to, you can even move the cursor to any line in the old display and use it to replace bytes. Of course, you have to be in the R command and have the store mode armed.

A little practice with the Screen Editor commands will pay off handsomely for you. They are very powerful -- better than provided on most large computers.

SYNTACTIC ELEMENTS

hex-address - Memory address of bytes to be displayed.

This parameter is the location in memory of the first of the eight bytes which will be displayed on the screen.

EXAMPLES

```
VR 5000<RETURN>
```

results in the display of the bytes at memory location 5000 hex.
Typing:

```
X<RETURN>
```

arms the store mode to allow you to change bytes between 5000 and 5007 hex.

SYNTAX -- S [set-specification]...

PURPOSE -- The Set/Status Command enables you to customize many features of TOP-DOS to your own needs, including selecting options and specifying the configuration of your disk drives. This command is also used to get the Status Display and to initialize the disk buffers.

DESCRIPTION -- The S command allows you to set a number of options and disk-drive configuration parameters. It accepts any number of specifications. After handling all the specifications, it initializes disk buffers and MEMLO, if appropriate, and finally puts the Status Display on the screen.

Initializing the disk buffers includes interrogating all the disk drives on the drive list, to determine their density, and allocating buffers for the drives and for open files.

HINT: To make any change permanent, write out TOP-DOS to a disk, using the H command. Then, when you boot that disk, TOP-DOS will come up with the options set as you want.

The parameter prompt for the S command, which is displayed if you just type "SRETURN>", is the ten-line menu:

<P>	Prompt Char.	<J>	Just Status
<L#>	Left Margin	<I>	Initialize
<S#>	System Drive#	<A>	Auto-RS232
<N#>	Num Buffers	<a>	Clear Auto
<D#>	Add Drive#		Bypass Cart.
<D#R>	Remove Drive#		Clear Bypass
<D#S>	Single Density	<R>	Resident DOS
<D#D>	Double Density	<r>	Clear Res.
<D#Q>	Quad (DS,DD)	<V>	Verify Write
<D#M>	Modify Bytes	<v>	Clear Verify

Since this prompt occupies almost half the screen, you may want to avoid getting it, if you don't need it, in order to leave more of your work history on the screen. So, use the one-line command format, when you don't need this menu.

NOTE: The disk buffers will be initialized and MEMLO reset, if any one of the S command parameters, N, D, I, or R, is specified.

CAUTION: The RS232 handler may be destroyed, if the disk buffers are initialized (as determined in the above note) -- unless the DOS-Resident option was ON when TOP-DOS was Booted. (The problem with the RS232 handler is that it loads in at, and changes, MEMLO. But initializing the disk buffers also changes MEMLO.)

Therefore, after calling the S command with N, D, I, or R to re-configure your disks, etc., re-boot before attempting to use the RS232 handler.

SYNTACTIC ELEMENTS

set-specification... [J] - The specification of one set action.

This parameter specifies one action for the S command to take. The "... " means that any number of such parameters may be given, each separated by a Comma ", ". If no parameter is given, the default is the J parameter (described below).

NOTE: When specifying a disk drive number, you may use either "D#" or just "#", where "#" represents a number from 1 to 8.

Each parameter consists of one of the following:

Pc - Change the Command Prompt character to "c", where "c" represents any character. As delivered, this character is set to ♥ (which is obtained by typing "<CTRL>-").

L# - Change Left Margin to "#", a small decimal number, normally 0 or 2. (The change will not be seen until after you execute the next command.) Any number larger than 2 may cause TOP-DOS's display lines to wrap-around the screen, making them hard to read. As delivered, the Left Margin is set to 0, to give you the full screen. Set it to 1 or 2, if your TV "overscan" causes you to lose the first or second characters on the line when it is set to 0.

S# - Change System Drive to Drive-#, where "#" is a number from 1 to 8. The System Drive is normally Drive-1.

N# - Change Number of open-file buffers to "#", a number from 1 to 16. Each single-density open file requires one such buffer and each double-density, two. As delivered, the number of buffers is set at 8.

D# - Add Drive-# ("#" is a number from 1 to 8) to the drive list. As delivered, Drive-1 and Drive-2 are in the drive list. If you have more than two drives, you should add your other drive numbers to the drive list.

D#R - Remove Drive-# from the drive list. If you only have one drive, you should remove Drive-2 from the list, as delivered.

D#S - Set Drive-# to Single-density. Use this only if your drive is software switchable and PERCOM compatible.

D#D - Set Drive-# to Double-density. Use this only if your drive is software switchable and PERCOM compatible.

D#Q - Set Drive-# to "Quad" density (double-sided double-density). Use this only if your drive is software switchable, PERCOM compatible, and double-sided.

D## - Modify Drive-# control bytes. This is only for the ADVANCED USER who understands the 12-byte PERCOM1 protocol. It causes the 12 bytes to be read from the drive and displays them on the screen, using the R command format. (Actually, 16 bytes are displayed: ignore the last 4 bytes.) Change any of the 12 bytes, using the method described in the R command. Then type "`^/RETURN`" to exit from the R command mechanism. The modified 12-byte protocol is then written back to the drive.

J - Just display Status Display. Use this if you just want to get the Status Display, but do not want to change any options.

I - Initialize the disk buffers. The effect of initializing the disk buffers is described above. Use this parameter if you just want to accomplish this function.

A - Turn ON the Automatic RS232-MEM.SAV option. This option automatically turns on the MEM.SAV facility if needed when the RS232 handler is loaded from the ATARI 850 Interface Module. This option is ON, as delivered, and is fully described in Chapter Six.

a - Turn OFF the above A option.

B - Turn ON the Bypass-cartridge option. This option causes a cartridge, like BASIC, to be bypassed on Boot-up, so that the system will go directly to the TOP-DOS Menu. This option is OFF, as delivered.

b - Turn OFF the above B option.

R - Turn ON the DOS-Resident option. This option causes all of TOP-DOS to remain in memory when you go to a user program, such as BASIC. This allows you to go back and forth between BASIC and the TOP-DOS Menu commands, without losing your BASIC program in the process and without the need to use the MEM.SAV facility. This option is OFF, as delivered, and is fully described in Chapter Six.

r - Turn OFF the above R option.

V - Set the disk write command to write-with-Verify. The Verify action consists of reading back the data written on a disk to insure its validity. This option is OFF, as delivered, since it slows disk writes and it is felt that it is not needed to insure data integrity.

v - Turn OFF the above V option.

EXAMPLES

To get started, type:

```
♥S<Return>
```

to get the S command menu. You'll need this menu until you are more familiar with all the many parameter choices.

After the menu displays, type in one or more choices, each separated by a Comma ",". If no parameter is given, you will just get the Status Display.

The most frequently-used call to the S command will probably be:

```
♥S J<RETURN>
```

which will just give the Status Display, but make no changes to the options or drive configuration, and will not initialize the disk buffers. Another frequently-used command is:

```
♥S I<RETURN>
```

which you would use to initialize the disk buffers after connecting, disconnecting, turning ON or OFF any disk drive, or manually switching its density.

```
♥S P>,L2,S2<RETURN>
```

sets the Command Prompt to ">", sets the left margin to 2, and sets the System Drive to Drive-2.

```
♥S N10,D2R,D3<RETURN>
```

sets the number of buffers for open files to 10, removes Drive-2 from the drive list, and adds Drive-3 to the drive list. Drive-2 will no longer be accessible, even if it is connected and ON.

```
♥S D1S,D2D,D3Q<RETURN>
```

sets Drive-1 to single-density, Drive-2 to double-density, and Drive-3 to double-sided double-density ("quad" density). This example assumes that these drives are software switchable and PERCOM compatible.

As noted above, the initial "D" is optional when specifying a disk drive in the S command. Thus, the above example could be written:

```
♥S 1S,2D,3Q<RETURN>
```

to accomplish the same function and is easier to type and read.

```
♥S a,B,R,U<RETURN>
```

turns the Automatic-RS232 option OFF, the Bypass-Cartridge option ON, the DOS-Resident option ON, and the Verify-Write option ON.

SYNTAX -- T [error#]

PURPOSE -- The primary purpose of this command is to interpret an error number that you may encounter when using TOP-DOS commands. This saves having to look up error code numbers.

DESCRIPTION -- The T command displays a message explaining an error number. The error number is either the one entered by the user as the parameter to the command or the one displayed in an "ERR ###" message.

To interpret the error number in an "ERR ###" message, just type "T<RETURN>" and TOP-DOS displays a brief explanation of that error.

To permit this simple quick method of interpreting a displayed error number, the T command has no parameter prompt. If it did have a parameter prompt, then when you typed "T<RETURN>", you would get the parameter prompt rather than the error message.

Error numbers (in contrast with descriptive messages) are frequently used in small computers because messages require a lot of memory space to store.

SYNTACTIC ELEMENTS

error# [last-err#] - The error number to be explained.

This parameter is the error number you want interpreted. If not given, the parameter defaults to the last error number displayed.

EXAMPLES

If you try to delete a locked file, you will get:

ERR 167

Typing:

▼T<RETURN>

will give you:

File locked

Similarly, attempting to write to a write-protected disk, results in:

ERR 144
▼T<RETURN>
Device "Done" (Not Ready)

(This message is not specific because this error could be caused by a number of conditions.)

It may take a little time to determine the problem, but this system is better than many which either give you no clue at all, or just say "I/O Error".

SYNTAX -- U filespec[/N]

PURPOSE -- The Undelete command is used to restore files that have been deleted. It can also recover files and sectors tied up in improperly closed files.

DESCRIPTION -- TOP-DOS searches the File Directory for the files in the filespec marked as 'deleted' or 'open'. For each file found, you are queried (unless the /N option is used):

Undelete D1:OLD_FILE.TXT?

If you answer "Y<RETURN>", the file will be restored. If you answer "N<RETURN>" or just <RETURN>, the file will be skipped.

NOTE: If you have written to the disk since the file in question was deleted, some or all of the sectors of the deleted file may have been reused and therefore not recoverable for the deleted file. In this case, an "ERR 164" (Attempt to Take a Used Sector) will be displayed. However, the deleted file will be restored in the File Directory, with as many sectors as were not reused. Thus, you may be able to recover part of your deleted file, if some, but not all, of its sectors were reused.

How to See the Deleted Files in the Directory

Use the /B or /D options in the A command to list the deleted and open files in the Directory listing. Deleted files are indicated in the listing by a "-" before the filename and open files by "!".

What if the File Already Exists?

If a "current" (that is, not deleted or open) file of the same name already exists, you will be told:

File exists!

and the file will be skipped. If you want to restore the deleted file, you must rename the existing file, at least temporarily. Then you can restore the deleted file.

What if there are Two or More Deleted Files of the Same Name?

Not too likely, but it can occur. In this case, undelete the first. Then rename it and undelete the next, etc.

How to Handle "Open" Files

The Undelete command can handle "open" files, if you find them in your Directory when you get a Directory listing with the A command using a /B or /D option. Such files were not properly closed for some generally unknown reason. These files take up slots in the File Directory and may also have sectors connected, which make them unavailable for use. Other DOS's will not access them.

However, this TOP-DOS U command allows you to access and properly dispose of these open files! Operate on these files with the U command as though they were deleted files, to bring them back into the "real world". Then delete them with the D command to free up their sectors and file slots. In the process it is normal to get an error message, such as "ERR 163" (Attempt to Free a Free Sector).

SYNTACTIC ELEMENTS

filespec [D1:X.X] - The specification of the files you want restored.

This parameter specifies the files to be undeleted. Wildcards may be used in the filespec, just as they may in the D Delete command.

/N - No Query for Undelete.

This option suppresses the query for undelete. TOP-DOS will undelete each deleted or open file as it is found in the search. Use this option to save time when you want to restore all deleted and open files specified in the filespec.

EXAMPLES

♥U OLD_FILE.TXT<RETURN>

will restore the file, assuming it is in the Directory marked as deleted or open, and assuming that a file of the same name does not already exist as a current file, and assuming that you answer "Y<RETURN>" when queried for permission to undelete.

♥U *.TXT/N<RETURN>

will restore all deleted or open .TXT files without queries, if files of the same names do not already exist.

CHAPTER SIX FEATURES AND OPTIONS

This chapter covers the features and options of TOP-DOS not previously covered (except for the MEM.SAV feature described in Appendix B).

AUTORUN FILE FACILITY

This feature provides for the automatic execution of a binary file on boot-up. If the disk in Drive-1 contains a binary file with an AUT extension, it will be loaded and executed after the DOS.SYS file is loaded into memory. If you have a BASIC or other cartridge inserted, the .AUT file is executed before control is passed to the cartridge.

This feature differs from a similar feature of the ATARI DOS 2.0. The ATARI DOS will only execute a file named AUTORUN.SYS, whereas TOP-DOS will execute any file with an AUT extension. This flexibility in TOP-DOS allows you to have a number of binary files on a disk that you may want to use as Autorun files, at different times. Each file can have a unique filename, which it retains when you activate it by renaming it with an AUT extension.

You can make an Autorun file out of any binary file that has a Run address by using the E command to rename it with an AUT extension. For example:

```
ⓂE MEDIT,MEDIT.AUT
```

or (to show you a shortcut):

```
ⓂE MEDIT,X.AUT
```

will setup the ATARI MEDIT editor as an autorun file. The next time you boot up the disk containing this file, it will be automatically executed, without your having to load it with the L command. This allows you to insert a disk containing MEDIT.AUT and text files, turn ON your ATARI, and go off to perform some other chore. When you return, MEDIT will be ready for you to use!

NOTE: If you have more than one file with an AUT extension, the first one in the File Directory will be used as the Autorun file.

RS232 AUTORUN FILE

One of the files included on your TOP-DOS disk is an Autorun file named RS232.TOP. By renaming this file to RS232.AUT, it will load the RS-232 handlers from an ATARI B50 Interface Module. This is needed if you are going to use your ATARI for telecommunication over the telephone lines, using a modem. (If you are not following this presentation, you probably don't need this information now. Skip to the next section.)

For those familiar with the ATARI DOS 2.0, the RS232.TOP file is equivalent to the AUTORUN.SYS file supplied with that DOS, with one notable exception: The RS232.TOP file patches the RS232 handler so that it is not destroyed by <SYSTEM-RESET>.

HELLO FACILITY

The HELLO feature of TOP-DOS provides for the automatic execution of a command file on Boot-up. If the disk in Drive-1 contains a file named HELLO (with a null extension), it will be executed as a command file just after the command Menu is displayed at Boot-up.

This feature allows you customize your disks with a message and/or a set of TOP-DOS commands, when that disk is booted. Or you can create a common HELLO file to put on all your disks to initialize your system when you boot it up. For example:

```
♥S 20,3Q<RETURN>
```

in a HELLO file will set Drive-2 to double-density and Drive-3 to "quad"-density, when it is executed on Boot-up.

A HELLO file is created just like any other command file. (See the Q command Chapter Five).

NOTE: A convient way to deactivate a HELLO file is just to rename it. If you have only one HELLO file on a disk, you could rename it by adding an extension. For example:

```
♥E HELLO,HELLO.SET
```

will rename it to HELLO.SET. (The extension SET is arbitrary, any legal extension will do.) However, if you want to maintain a number of HELLO files on a disk, then you might use the extension HEL to hold inactive ones, using the filename to identify the purpose of the file. For example:

```
♥E HELLO,Set_Dens.HEL
```

will deactivate the HELLO file, leaving a filename to remind you what's in it.

OOS-RESIDENT FEATURE

This feature allows you to have all of TOP-OOS always in memory, so you can go back and forth between BASIC, or other user program, and the TOP-OOS Menu commands, without the timeconsuming MEM.SAV disk file transfers, which are otherwise necessary.

This feature is accomplished by TOP-DOS advancing the address in MEMLO to beyond the end of the DUP-Area (see Appendix A). Programs which properly use MEMLO, then, will not conflict with TOP-DOS's use of this area.

The cost of this feature is about 9K to 18K bytes of program space. For a program like BASIC or an editor or wordprocessor, this means a smaller buffer area. But for some programs which load into low memory area, there may be an overlap with the DUP-Area. In this case, this feature is not applicable.

One way to determine if a program loads over part of the OUP-Area is to load the binary program in question, using the /M and /N options, the latter to inhibit execution. For example, if the program is the ATARI MEDIT editor:

```
♥L MEDIT/M/N<RETURN>
```

will display the memory area(s) loaded by the MEDIT file. If there is an overlap of the DUP-Area, after the load is complete, the DUP.SYS file will be automatically reloaded and the its area will be displayed. (For this test, the MEM.SAV feature should be OFF.)

Note, for this example, there is a large overlap. MEDIT starts at 2600 hex, while OUP occupies memory up to about 4800 hex.

Use this DOS-Resident feature when there is no overlap of your programs with the DUP-Area and when the loss of buffer space is not a problem.

To turn this DOS-Resident feature ON, use the R parameter of the S command. To turn it OFF, use the "r" parameter.

AUTO-RS232 FEATURE

The RS232 handler loads in from the ATARI 850 Interface Module starting at the memory location contained in MEMLO. Thus, unless the DOS-Resident feature is ON, the RS232 handler will lie in the OUP-Area. Then, when OUP.SYS loads into the DUP-Area to bring you the Menu commands, it will overwrite the RS232 handler -- unless, of course, the MEM.SAV feature is ON! (See Appendix A for a discussion of MEMLO and the DUP-Area, and Appendix B, for the MEM.SAV facility.)

Since the RS232 handler is loaded by the Autorun file RS232.AUT at Boot-up, the MEM.SAV facility must already be ON or the RS232 handler will be overwritten and destroyed by DUP.SYS loading into memory. You could, of course, have anticipated this problem by having turned ON the MEM.SAV facility and then having written TOP-OOS files to your disk. In that case, the MEM.SAV facility would be ON at Boot-up and the RS232 handler would be protected.

But, in case you didn't prepare for this problem, TOP-OOS provides the AUTO-RS232 facility, which turns on MEM.SAV when needed to protect the RS232 handler from being overwritten. Actually, this facility is more general and will protect any program which is loaded by an .AUT file at Boot-up.

This AUTO-RS232 facility is ON, as delivered. It may be turned OFF by the "a" parameter of the S command. Turn it OFF if you use a telecommunication program that runs in BASIC, do not need the RS232 handler preserved when you go to the TOP-OOS Menu commands, and do not want the MEM.SAV turned ON, with the resulting disk-file operations.

APPENDIX A STRUCTURE OF TOP-DOS

This appendix, for the **ADVANCED USER**, describes the structure of TOP-DOS on disk and in memory. It includes the memory map of the programs, storage, and buffers of the TOP-DOS system.

For maximum compatibility with the ATARI DOS 2.0, the structure and nomenclature of TOP-DOS has been kept close to that of the ATARI DOS.

OS vs DOS

First, let's clear up a common confusion between an OS (Operating System) and a DOS (Disk Operating System). As will be seen, these are two different, although related, systems programs. Understanding some of their interactions is helpful in understanding the role and function of a DOS and, in particular, of TOP-DOS.

An OS is the software that makes a computer generally accessible and usable. It provides input/output facilities so you can type in commands, see what you type on the screen, output to a printer, boot-up a disk, etc. In the ATARI computers, the OS is contained in ROM (Read-Only Memory), so it is always available when you turn ON your computer.

A DOS is the software that provides the access to your disk drive units. It organizes the sectors on a disk into files, maintains and interprets a File Directory on a disk, and provides you with the facilities and commands to work with disks and the files they contain.

TOP-DOS (like ATARI DOS 2.0) consists of two parts:

1. DOS: The memory-resident part that must remain in memory.
2. DUP: The part that provides the Menu commands.

The first part, DOS, is contained in the DOS.SYS file and the second, DUP, in the DUP.SYS file. (DUP stands for Disk Utility Program, the terminology established by the ATARI DOS.)

There is an interaction between an OS and a DOS. On the ATARI computers, the OS is designed to function without a disk, and therefore without a DOS. You can run cartridge programs to play games, using just the OS. If you write programs in BASIC or Assembler, or if you run a program which generates data, you need a way of saving information when you turn OFF your computer. If you use a cassette recorder to store programs or data, OS has provision to handle the necessary operations.

However, when you get a disk drive to store your information, a DOS is required, due to the much more complex function of organizing the disk into files. Although the DOS routines could have been incorporated in with the OS ROM, that would have greatly increased its size in memory, leaving little room for user programs. Instead, ATARI wisely provided to keep the DOS separate, residing on disk, to be loaded into RAM when needed for disk operations.

THE BOOT PROCESS

When you turn ON your computer, the OS checks to see if you have a disk Drive-1 connected and ON. If so, the OS reads in Sector-1 of the disk in Drive-1 to a buffer in low memory. From this sector, the OS determines the number of sectors to be read and the location in memory to which they are to be read.

This is called a "bootstrap" operation (or "boot", for short), and comes from the phrase "lifting yourself by your bootstraps". The sectors read by the OS are called "boot" sectors.

In TOP-DOS, as in the ATARI DOS 2.0, there are three boot sectors. The small program in these sectors is loaded by the OS at location 0700 hex in memory. This small program then "boots" in the rest of the first part of TOP-DOS from the file DOS.SYS. TOP-DOS then loads and executes an .AUT file, if one exists on Drive-1.

At this point, if a cartridge is present, it is executed. If not, TOP-DOS reads in DUP from the DUP.SYS file, which contains the Menu and the routines to execute the Menu commands.

MEMORY MAP OF TOP-DOS

Figure 1 shows the memory areas occupied by the two parts of TOP-DOS, DOS and DUP. As is seen, these parts are separated in memory by an area for disk buffers. It is also seen that the DUP program has an associated memory area from 2480 to 2680 for storage. This storage area, together with the program area of DUP is referred to as the "DUP-Area", and it is this DUP-Area which is shared with user programs (see Appendix B, The MEM.SAV Facility).

NOTE: The areas shown in Figure 1 are not to scale. Also, the end of the DUP-Area, shown as 4800 hex, is approximate and may change with later releases of TOP-DOS. For a given release, you can determine this address by turning ON the DOS-Resident option using the R parameter of the S command and looking at the value of MEMLO in the Status Display. Also, this address is stored at 1550 hex (low-order byte first, in keeping with the method of storing 6502 addresses).

As mentioned above, DOS must remain in memory, even when BASIC or other user program is run. However, DUP is only needed when you are working with the TOP-DOS Menu commands. Therefore, the DUP-Area may be used by BASIC or other user program. When returning to TOP-DOS from BASIC (by typing "DOS") or other user program, if any bit of any byte of the DUP-Area has been changed by the user program, TOP-DOS will re-load DUP from the DUP.SYS disk file.

NOTE: To insure the integrity of DUP in memory, and to detect any change in the DUP-Area by a user program, a three-byte check-sum is computed when leaving TOP-DOS to run a user program. When returning to TOP-DOS, this check-sum is recomputed and compared with the previous value. This three-byte check-sum provides a 16,000,000-to-1-odds assurance of the detection of any change in the DUP-Area!

DISK BUFFERS

As seen in Figure 1 of Appendix A, the space between DOS and the DUP-Area is left open for disk buffers. (A disk buffer is an area of memory used to hold the data of a disk sector, during the process of reading or writing it to or from a disk.) This space is large enough to hold 28 128-byte buffers.

Disk buffers are allocated by the initialization routine of TOP-DOS. This routine is called when TOP-DOS is booted at power-up. It is also called on <SYSTEM-RESET> and by the S command (see Chapter Five).

During this initialization process, each disk drive in the drive list is interrogated to determine its status (active?) and its density. One buffer is allocated for each active single-density drive and two for each double-density drive. (Remember that a single-density sector holds 128 bytes and a double-density, 256 bytes.) These buffers for the active drives are used to hold the VTOC's of the disks.

The initialization continues the disk buffer allocation by adding the buffers for open files, according to the number given by the N parameter of the S command. This number is shown in the Status Display before "#8Bufs".

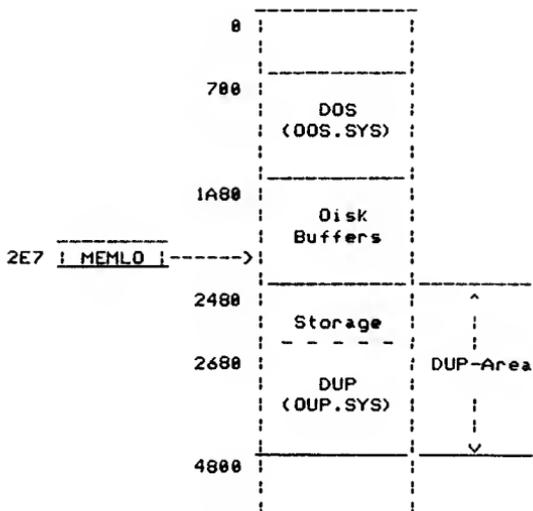
MEMLO

After the buffers are allocated, the memory address just beyond the last buffer is stored at MEMLO. MEMLO, established by the OS at 2E7 hex (743 decimal), holds the address of the first byte of memory available for user programs. If total number of buffers for both drives and open files is equal to 28, then MEMLO would contain 2480 hex, which is the beginning of the DUP-Area. (28 times 128 = 2560 or A00 hex; 1A80 + A00 = 2480 hex)

If fewer than 28 buffers are allocated, MEMLO would point to an address less than 2480 hex. It is doubtful that you would need more than 28 buffers, but if you did, MEMLO would point to an address larger than 2480.

If the DOS-Resident option is ON, TOP-DOS will advance the address in MEMLO to just beyond the DUP-Area, no matter how many buffers are allocated. (See Chapter Six for more about the DOS-Resident feature.)

Figure 1. Memory Map of TOP-DOS.



NOTE: Addresses are in hex.

APPENDIX B THE MEM.SAV FACILITY

This appendix assumes familiarity with Appendix A and is for the ADVANCED USER. It describes how the MEM.SAV facility works, not just how to operate it. An understanding of this appendix should help you:

- o Know when to use this facility and how it will help you.
- o Know when you can avoid using it, to save the time it takes for its disk operations.
- o Know why it is reading or writing the MEM.SAV file under different circumstances.

SHARING MEMORY BY DUP AND USER PROGRAMS

The DUP-area can be shared by DUP and user programs in two ways:

1. By restoring DUP from DUP.SYS when going to the DUP Menu from a user program. In this case, the program's data in the DUP-area is lost.
2. By swapping the program's data to and from disk (MEM.SAV file), when going between the TOP-DOS Menu and your program.

SWAPPING DUP-AREA MEMORY

If a program uses the DUP-area, and you want to be able to return to your program after going to the TOP-DOS Menu, then the data in the DUP-area must be saved before DUP is read in over it.

In this case, the program's data in the DUP-area is written to the MEM.SAV file when you go to the TOP-DOS Menu and read back to the DUP-area when you return to your program.

This function is also performed when you load programs into the DUP-area or write binary files from it.

FACTORS CONTROLLING MEM.SAV ACTIONS

There are several factors that affect the action the MEM.SAV facility will take. These include:

- o Is the MEM.SAV facility ON?
- o Is the MEM.SAV file data valid?
- o Has a program written data in the DUP-area?
- o Has the data in the MEM.SAV file been loaded into the DUP-area?
- o Is a program about to be written to or from the DUP-area?

Let us look at several cases involving these factors.

1. A PROGRAM WRITES IN THE DUP-AREA

The first case to be considered is when a program writes into the DUP-area. The Atari BASIC cartridge is such a program and will be used as the example for this case. BASIC uses the DUP-area for storing the user program.

EXCEPTION: If the DOS-Resident option is enabled, the DUP-area is not used by programs, like BASIC, which correctly use MEMLO to determine where to begin their storage area. (See the DOS-Resident section of Chapter Six and also Appendix A.)

The action taken in Case-1 depends on several factors, as will be seen below.

IF MEM.SAV FACILITY IS OFF

If MEM.SAV is OFF, when you go to the TOP-DOS Menu from BASIC (by typing "DOS"), your program in the DUP-area is not saved on disk. Consequently, your BASIC program will be destroyed when the TOP-DOS Menu program (DUP) is restored to the DUP-area from the DUP.SYS file.

If you had SAVE'd your BASIC program before you left BASIC and if you don't plan to return to your BASIC program from the Menu, then you have no problem: it is OK that the MEM.SAV facility was OFF.

However, if you want to find your program intact when you return to BASIC, then MEM.SAV should be ON.

IF MEM.SAV FACILITY IS ON

If MEM.SAV is ON when you go to the TOP-DOS Menu from BASIC, your program in the DUP-area is saved on disk in the MEM.SAV file, before DUP is restored from the DUP.SYS file.

IF MEM.SAV FILE HOLDS VALID DATA

In this case, where MEM.SAV was ON and your program in the DUP-area was written to the MEM.SAV file on disk, TOP-DOS takes note that the MEM.SAV file holds valid data (<VALID> is shown in the Status Display).

When you return to BASIC from the Menu (by typing "B"), TOP-DOS loads the MEM.SAV file into the DUP-area.

Back in BASIC, your program is intact.

IF MEM.SAV FILE DATA NOT VALID

Suppose you decide that you don't want your BASIC program restored when going back to BASIC. Or maybe you are going to load and run another program, so that the data in the MEM.SAV file is no longer valid.

In this case, you can notify TOP-DOS to inhibit the reading and writing of the MEM.SAV file either by turning the MEM.SAV facility OFF:

▼N F<RETURN>

or by "clearing" the MEM.SAV facility:

▼N C<RETURN>

which leaves the MEM.SAV facility ON, but notes that the MEM.SAV file does not hold valid data (<INVALID> is shown in the Status Display).

WHY NOT LEAVE THE MEM.SAV FACILITY ON

Before continuing to the next case, let's answer the question: If the MEM.SAV facility is so great, why not leave it ON all the time?

The reason is simple: It takes a long time -- about 30 seconds -- to save and restore the DUP-area to and from the MEM.SAV disk file.

So when you don't need it, you'll want to turn OFF the MEM.SAV facility.

If you need the MEM.SAV facility ON, but want to stop it from saving and restoring invalid data, use the C Clear parameter in the N MEM.SAV command, as discussed above. That will stop the wasted operations, but leave the facility ON.

2. LOADING A PROGRAM TO THE DUP-AREA

The second case is where you load -- but do not run -- a binary program that overlaps the DUP-area.

PROGRAM, NOT DATA, IN THE DUP-AREA

Note that this case is different from Case-1.

In Case-1, the program did not itself overlap the DUP-area, but rather it stored data (your BASIC program, for example) in the DUP-area, when it ran.

In this case, Case-2, the program itself is loaded into the DUP-area (by the L command). This presents a special problem to TOP-DOS, since DUP gets overwritten in the process!

HOW TOP-DOS LOADS TO THE DUP-AREA

You might wonder how, when you are in the TOP-DOS Menu, you can use the L Load command to load a file to the DUP-area, where DUP itself resides.

The answer is that the loader is not in the DUP-area, but rather in the DOS (memory-resident) area. All TOP-DOS does for an L command is to get the file information from you and then transfer control to the memory-resident (DOS) part of TOP-DOS to perform the load.

If the binary file you are loading contains data to be loaded in the DUP-area, it will destroy the TOP-DOS Menu program in the process. TOP-DOS takes care of this by reloading DUP.SYS into the DUP-area before returning to the TOP-DOS Menu.

IF MEM.SAV FILE HOLDS VALID DATA

If the MEM.SAV file holds valid data, then it will be loaded into the DUP-area before a binary program which overlaps the DUP-area is loaded.

The reason the MEM.SAV file is loaded before the binary file is that the MEM.SAV file may hold data in a different part of the DUP-area than that used by the binary file. If the MEM.SAV file were not loaded first, the data in the MEM.SAV file would be lost when the DUP-area is later written to the MEM.SAV file.

AFTER THE LOAD, BACK TO THE MENU

After the binary load is completed, we must get back to the TOP-DOS menu. (Remember, in this case, Case-2, the program is loaded -- but not run. So we have to go back to the Menu.) To get back to the Menu, DUP must be restored to the DUP-area from the DUP.SYS file. In the process, the binary program just loaded will be overwritten and destroyed. To avoid the loss of the program data in the DUP-area, it must first be saved in the MEM.SAV file.

Thus, the MEM.SAV facility must be ON before loading a binary file into the DUP-area, unless the binary file is to be run at its Run address, as part of the L Load process (see Case-3).

RECAP OF CASE-2

Loading -- but not running -- a binary file which overlaps the DUP-area requires the following steps:

- o The MEM.SAV file is loaded into the DUP-area, if the MEM.SAV file holds valid data and the MEM.SAV facility is ON.
- o The binary file is loaded.
- o The DUP-area is saved to the MEM.SAV file, if the MEM.SAV facility is ON.
- o DUP is restored to the DUP-area from the DUP.SYS file.
- o The TOP-DOS Menu program is reentered.

3. LOADING AND RUNNING A PROGRAM IN THE DUP-AREA

This case is related to Case-2.

In Case-2, the program in the DUP-area is not run. In this case, Case-3, the program is run, using the Run-Address contained in the binary file.

This difference is important, since in this case, we do not have to return to the TOP-DOS Menu after the load. Thus, the DUP-area does not have to be saved to the MEM.SAV file and DUP does not have to be restored from the DUP.SYS file!

However, if the MEM.SAV file holds valid data, the DUP-area will still be restored from the MEM.SAV file before the binary file is loaded into the DUP-area.

RE-CAP OF CASE-3

The following steps are taken in Case-3, where a binary file that overlaps the DUP-area is loaded and run:

- o The DUP-area is restored from the MEM.SAV file, if the MEM.SAV file holds valid data.
- o The binary file is loaded.
- o The program is run.

NOTE: Case-3 is simpler than Case-2, because we do not return to the TOP-DOS Menu after the load, since the program is immediately run. Thus, the TOP-DOS Menu does not have to be restored or the program in the OUP-area saved.

4. RUNNING A PROGRAM NOT IN THE DUP-AREA

Before running a program, even if not in the OUP-area, the MEM.SAV file will be loaded to the OUP-area, if it holds valid data.

To understand why this is done, it is important to realize that the OUP-Area is considered to be part of the user-program memory. When you are using the TOP-DOS Menu commands, OUP (from the OUP.SYS file) is in the OUP-Area. The MEM.SAV file represents the user-program "image" of the OUP-Area at this time, if it holds valid data. Therefore, when going to a user program (by the M or P commands), this user-program image of the OUP-Area is restored from the MEM.SAV file.

5. BINARY SAVE FROM THE DUP-AREA

This case is where the area to be written by a Binary Save "K" command overlaps the DUP-area.

If the MEM.SAV file holds valid data, this file will be loaded to the OUP-area, before the binary file is written.

HOW TO TURN ON MEM.SAV WHILE IN BASIC OR ASSEMBLER

If you find yourself in BASIC (or ASSEMBLER) with the MEM.SAV facility OFF, you can turn it ON by:

```
POKE 5445,128   [BASIC]
C 1545<B0      [ASSEMBLER "BUG"]
```

(Or you can turn the facility OFF by POKE'ing a 0 in the same location.)

SIZE OF THE MEM.SAV FILE

The MEM.SAV file is 71 sectors long for the first release version of TOP-DOS. (Later releases may increase the size if new features are added.) Room for this file must exist on the disk in Drive-1, to use the MEM.SAV facility.

APPENDIX C
HEXADECIMAL NOTATION

This appendix presents a brief explanation of the hexadecimal notation ("hex", for short). A table is given to aid in the conversion between decimal and hex.

The use of the hex number system stems from the extensive use of the 8-bit "byte" as the basic unit of information in computers. To aid in representing its bits, a byte is divided into two 4-bit parts called "nibbles".

Each nibble can assume one of 16 values from 0 through 15. Thus, a hexadecimal, or 16-based, number system follows as the natural method of representing data and addresses in computers. Each nibble is represented by one hex digit; a byte, by two hex digits; a two-byte "word" or address, by four hex digits, etc. (Words and addresses in the 6502 are two bytes in size.)

In the decimal number system there are ten values to be represented: 0 through 9. Conveniently, there are ten numerals, 0 through 9, to represent these ten values.

However, since widespread use of the hexadecimal number system is relatively new, there are no numeral symbols to represent the six values above 9. Therefore, six symbols had to be chosen to represent these values. The six so chosen are the capital letters A through F. Thus, the 16 hex digits are 0 through 9 and A through F (A=10, B=11, C=12, D=13, E=14, F=15).

The basic principle of a number system is that each digit in a number carries the weight of its position in the number. For example, in the decimal system, the value of the number:

2345

can be determined by adding the contribution of each digit. The low-order digit (the right-most one) carries the weight of 1; the next digit, of 10; the next, of 100; the next, of 1000; etc. Thus, for this example:

$$\begin{array}{r} 5 \times 1 = 5 \\ 4 \times 10 = 40 \\ 3 \times 100 = 300 \\ 2 \times 1000 = 2000 \\ \hline \text{Total} = 2345 \end{array}$$

In the hex number system, the same principle applies, but the base is 16, rather than 10. If the number in the above example were hex, it would have the value:

$$\begin{array}{r}
 5 \times 1 = 5 \\
 4 \times 16 = 64 \\
 3 \times 256 = 768 \\
 2 \times 4096 = 8192 \\
 \hline
 \text{Total} = 9029
 \end{array}$$

Thus, 2345 hex = 9029 decimal. Note that each of the weights is 16 times the size of the previous weight for the 16-based hex system, whereas, for the 10-based decimal system, the multiplier of the weights is 10.

For another example, the value of the hex number:

ABCD

is evaluated:

HEX	DECIMAL
D x 1 = 13 x 1 = 13	
C x 16 = 12 x 16 = 192	
B x 256 = 11 x 256 = 2816	
A x 4096 = 10 x 4096 = 40960	

Total	= 43981

Thus, ABCD hex = 43981 decimal.

The table given below helps in converting between hex and decimal. Up to four hex digits can be converted, which is all you will need for the ATARI computers.

To convert from hex to decimal, just add the weighted values from the columns of the table according to the position of the digits of a hex number.

To use the table in the reverse direction requires subtraction. Start with the decimal number and find the largest number in the table which is not larger than the number itself. Subtract the table number from the given number. Repeat the process with the difference until nothing is left.

Using the above example, backwards, convert the decimal number:

43981

to hex. From the table, the largest number not greater than 43981 is 40960, corresponding to the hex digit "A", in the 4th column. Then:

43981	-	40960	=	3021:	A000
3021	-	2816	=	205:	800
205	-	192	=	13:	C0
13	-	13	=	0:	D

Total					ABCD
-------	--	--	--	--	------

HEX-DECIMAL CONVERSION TABLE

HEX	DECIMAL			
	1ST	2ND	3RD	4TH
0	0	0	0	0
1	1	16	256	4096
2	2	32	512	8192
3	3	48	768	12288
4	4	64	1024	16384
5	5	80	1280	20480
6	6	96	1536	24576
7	7	112	1792	28672
8	8	128	2048	32768
9	9	144	2304	36864
A	10	160	2560	40960
B	11	176	2816	45056
C	12	192	3072	49152
D	13	208	3328	53248
E	14	224	3584	57344
F	15	240	3840	61440

APPENDIX D
TOPICS FOR ADVANCED USERS

This appendix contains topics of interest to advanced users who may wish to take advantage of some capabilities of TOP-DOS which are available to an assembly-language program. The reader of this appendix is assumed familiar with the 6502 microprocessor and the ATARI DS.

All locations and byte values are given in hex.

KEY LOCATIONS

0714 - Device Time-out
0715 - Device error retry count
1544 - Flag: MEM.SAV file holds valid data
1545 - Flag: MEM.SAV facility is ON
154C - Error number of last error
154F - Flag: DOS-Resident option is ON
1550 - Address of first location beyond DUP-Area (2 bytes)
1552 - Flag: Display memory map on binary load
1554 - Left margin
1555 - Flag: Bypass-Cartridge option is ON
155A - Default run address for P command (2 bytes)
2480 - Command line input buffer (80 decimal bytes)
24D0 - Command parameter buffer (40 decimal bytes)
24F8 - Command-line offset-pointer
24F9 - Parameter offset-pointer
2602 - Flag: Error in reading parameter
2680 - Subroutine: Get file name
2683 - Subroutine: Get file name & set IDCB
2686 - Subroutine: Get decimal number
2689 - Subroutine: Get hex number

- NOTES: 1. Flags: Set, if negative; Clear, if positive
2. Offset-pointer: Byte offset in a buffer
3. All locations contain one byte, unless otherwise noted.
4. Device Time-out is the time in seconds that TDP-DDS waits for a device to respond before trying again. The ATARI DDS 2.0 set this timeout at 15 (0F hex). TDP-DDS has reduced it to 2, to speed recovery. If you experience timeouts with your drives, you can increase this parameter.
5. Device error retry count is the number of times TDP-DDS will retry an operation before giving up and issuing an error report.

SUBROUTINES

The subroutines mentioned above and described below facilitate the access and interpretation of the command-line input. These subroutines take their input from the command line buffer and store their output in the parameter buffer, in the 6502 registers, or in an IDCB.

These subroutines assume that the command line is in the command-line buffer and that the command-line pointer contains the offset of the next byte in the command line to be interpreted. These conditions are in fact established by TOP-DOS.

Get File Name Subroutine

IN: Command-line pointer: Next byte in Command Line buffer
Parameter pointer: Where file name will be stored

OUT: Command-line pointer: Advanced beyond file name
Parameter pointer: Advanced beyond file name

Get File Name and Set IOCB Subroutine

IN: Command-line pointer: Next byte in Command Line buffer
Parameter pointer: Where file name will be stored
X: IOCB pointer

OUT: Command-line pointer: Advanced beyond file name
Parameter pointer: Advanced beyond file name
X: IOCB pointer
IOCB: Contains address of file name

Get Decimal Number Subroutine

IN: Command-line pointer: Next byte in Command Line buffer

OUT: A: Decimal number
Y: Terminating character of hex number
C: Set, if null (i.e.: no hex number was given)
Command-line pointer: Advanced beyond hex number
Error Flag: Set if error

Get Hex Number Subroutine

IN: Command-line pointer: Next byte in Command Line buffer

OUT: A: Low byte of hex number
X: High byte of hex number
Y: Terminating character of hex number
C: Set, if null (i.e.: no hex number was given)
Command-line pointer: Advanced beyond hex number
Error Flag: Set if error

NOTES: 1. The file name stored in the parameter buffer is terminated with an EOL (End Of Line) character.

2. The command-line pointer is advanced by these subroutines to beyond the terminating character, except if it is an EOL, in which case, the pointer points to the EOL.

3. These subroutines lie below 4800, so the program which calls them should be located above 4800.

The use of these subroutines is best illustrated with an example. Suppose you have written a program which reads an input text file and formats it into paragraphs. The output could be a disk file or the printer. After the program is written and assembled, it would be loaded with the L command:

```
L FORMAT<RETURN>
```

If used in this way, you would have to program the reading and interpretation of the input and output filespecs and other parameters. However, by using the above subroutines, you could simplify the programming task, as well as the use of the program. The program would then be called by:

```
L FORMAT,IN_TEXT.TXT,OUT_TEXT.FMT,10,70<RETURN>
```

where the parameters are given following the filespec of the binary load file. In this example, the parameters are the input filespec, the output filespec, the left margin, and the right margin. Of course, any names could be used for the input and output files, or the output filespec could be the Printer "P:". Your program would include:

```
LDX #10          USE IOCB-1 FOR INPUT
JSR $2683        GET INPUT FILE NAME AND SET IOCB-1
.
.               CODE TO OPEN THE INPUT FILE
.
LDX #20          USE IOCB-2 FOR OUTPUT
JSR $2683        GET OUTPUT FILE NAME AND SET IOCB-2
.
.               CODE TO OPEN THE OUTPUT FILE
.
JSR $2686        GET LEFT MARGIN (DECIMAL)
STA LEFT         AND STORE IT
JSR $2686        GET RIGHT MARGIN
STA RIGHT        AND STORE IT
.
.
```

After reading the parameters your program is then free to use the area below 4800 for storage, since the above subroutines will no longer be needed.

APPENDIX E
ERROR CODES

There are three sources of error messages: the ATARI OS, the DOS.SYS (memory-resident) part of TOP-DOS, and the DUP.SYS (Menu command) part of TOP-DOS. The first two use error numbers to conserve memory space. The last one gives brief messages.

This appendix lists the error numbers and messages for these three categories.

ATARI OS ERROR NUMBERS

The ATARI OS error numbers are listed here, with the interpretations from the ATARI BASIC Reference Manual and brief additional comments.

NO.	ATARI INTERPRETATION	COMMENTS
12B	<BREAK> Key	TOP-DOS interprets the <BREAK> Key as command interrupt, so it does not normally appear as an "error".
129	IOCB Already in Use	Probably a file was opened but not closed.
130	Nonexistent Device	Device letter not in handler table.
131	IOCB Open for Write Only	Attempt to read a file which was opened for write only.
132	Invalid Command	Illegal IOCB command code.
133	Device or File Not Open	Attempt to access a file that was not opened.
134	Bad IOCB Number	Program call to CIO did not set X to a legal IOCB pointer.
135	IOCB Open for Read Only	Attempt to write to a file which was opened for read only.
136	EOF (End of File)	Attempt to read beyond the end of the data in a file.
137	Truncated Record	The record being read is larger than the buffer used to read the record.
138	A Device Time-out	Device does not respond to computer (Possibly off-line).
139	Device NAK	Device responds to computer with a "Not Acknowledge" message, meaning it did not understand the request.
140	Serial Bus Frame Error	I/O communication garbled.
141	Cursor Out of Range	Cursor moved off screen edge.
142	Serial Bus Overrun	Computer did not respond fast enough to accept input data.
143	Serial Bus Checksum Error	I/O communication garbled.
144	Device Done Error	Probably attempt to write to a write-protected disk.
145	Illegal Screen Mode	Attempt to open Screen Editor with illegal mode number.
146	Function Not Implemented	Device handler does not support the IOCB command.
147	Insufficient RAM	Insufficient memory available for the screen mode selected.

TOP-DOS ERROR NUMBERS

The TOP-DOS error numbers are listed here, with interpretations and brief comments.

NO.	INTERPRETATION	COMMENTS
160	Bad drive#	The specified disk drive number is not available. Check the Status Display: it must show that drive number with either S, D, or Q density.
161	Too many open files	More buffers are needed for open files.
162	Disk full	All sectors of the disk are used.
163	Attempt to free a free sector	This may occur when a corrupted or "open" file is deleted.
164	Attempt to take a used sector	This may occur when a corrupted file is undeleted.
165	Bad filename	Filespec contains illegal characters.
166	Bad Point Data Length	The byte count in a POINT call is too large.
167	File locked	Attempt to delete a locked file.
168	Bad FMS command	FMS was called from a user program with an illegal command.
169	Directory full	The File Directory on a disk has no room for any more files.
170	Files not found	The specified file was not found in the File Directory.
171	Bad Point	Attempt to POINT beyond the end of a file or to a file not open for read.
172	DOS-1 Disk	Attempt to access a file written by ATARI DOS 1 (or could result from a corrupted disk). TOP-DOS does not handle files from the old ATARI DOS 1.
173	Bad sector	A bad sector was encountered during a disk formatting operation. The disk should be discarded.

TOP-DOS MENU-COMMAND ERRORS

The TOP-DOS Menu-command errors are listed here, with brief comments. These errors generally occur when calling Menu commands.

ERROR MESSAGE	COMMENTS
Bad Device	A device was used, which is illegal in the C command.
Bad Name	An improperly formed file name.
Bad Parameter	Error in parameter specification.
BRK Abort	6502 BRK instruction encountered.
Drives Incompatible	Disk drive densities differ for J Disk-Duplicate command.
Err# not listed	T command cannot find the error#.
IO ERR	I/O error during a binary load.
Invalid Hex	Illegal hex character.
Max=118 for Single-Density	I command parameter error.
Max=246	I command parameter error.
Need D1 thru D8	Error in disk-drive parameter. Use "D1" or "1", etc. (no ":").
No cartridge	8 command called with no cartridge inserted.
Not enough room	Inadequate memory space for J Disk Duplicate command.
Not TOPDOS Format	The specified operation can only be carried out with a TOPDOS disk.
Too Many Cmd. Files	The Q-command facility can only recurse to two levels deep.
Too Many Digits	More than four digits in a hex number is illegal.

TOP-DOS RAMDISK ENHANCEMENT

TOP-DOS SUPPORTS RAMDISK

If your TOP-DOS disk includes the files AXLON and/or MOSAIC, then it can be enhanced to handle a "ramdisk" with either the AXLON 128K RAM or the MOSAIC 64K SELECT RAM for the ATARI 800 computer.

In either case, the additional RAM is made to look like a disk drive to TOP-DOS.

ADVANTAGE OF RAMDISK

The advantage of having a ramdisk is the greatly increased transfer speed when reading or writing data.

Transfers between the ramdisk and RAM are almost instantaneous. Thus, you can load files from a ramdisk without the usual long delay imposed by a real disk load.

Transfers between the ramdisk and a real disk are about twice as fast as between two real disks.

AUGMENTING TOP-DOS FOR RAMDISK

In order to achieve this capability, TOP-DOS must be augmented with additional computer program code. This code is added to the memory-resident part of TOP-DOS -- that part contained in the DUS.SYS file. The additional code has been limited to one "page" (256 bytes). (See Appendix A of your TOP-DOS Manual for more information on the structure of TOP-DOS.)

The TOP-DOS contained on your disk is a "standard" TOP-DOS. That is, when you Boot the disk, TOP-DOS comes up without the ramdisk capability. You may use this TOP-DOS as is, or you may augment it with either of the ramdisks.

A SINGLE COMMAND CREATES YOUR RAMDISK

This augmentation of TOP-DOS to implement a ramdisk for either the AXLON or MOSAIC RAM is accomplished by a single TOP-DOS command. Simply load either the AXLON or the MOSAIC file:

▼L AXLON<RETURN>

or

▼L MOSAIC<RETURN>

This just takes a couple of seconds. TOP-DOS is now augmented with the ramdisk selected: AXLON or MOSAIC. The ramdisk is set to Drive-8, single-density.

CONFIRMING THE RAMDISK

Get a Status Display to see the effect of this augmentation process. (See the S command in your TOP-DOS Manual.)

♥S J<RETURN>

will produce the Status Display. It should now include "8S" in the drive configuration, that is, Drive-8, Single-density. It should also include either AXLON or MOSAIC to indicate that your TOP-DOS is augmented with that ramdisk capability.

Note that MEMLO has increased by 180 hex: 100 (256 decimal) for the addition page of code, plus 80 (128 decimal) for the buffer for your new ramdisk. (For a description of MEMLO, see your TOP-DOS Manual, pages 8 and 61. For a presentation of hex, refer to Appendix C.)

SAVING YOUR AUGMENTED TOP-DOS

Before using your new ramdisk, you might want to save the DOS files to another disk to preserve the augmented TOP-DOS. (See the H command in your TOP-DOS Manual for this operation.)

INITIALING YOUR NEW RAMDISK

To use your new ramdisk, first it must be initialized. (See the I command in your TOP-DOS Manual.) Always use the V option, since the ramdisk is not formatted like a real disk. (If you attempt to format a ramdisk, by omitting the V option, you will get an "ERR 139" (NAK) message.)

♥I 8<RETURN>

When you omit the second parameter, as shown in the above command, the ramdisk size defaults to the full use of all available RAM.

NOTE: If you have three MOSAIC 64K SELECT boards installed, you will have 144K of RAM for your ramdisk. But, TOP-DOS can only handle 118K for a single-density drive. In this case, the ramdisk will be initialized at its maximum size (118x8=944 sectors) and you will get the message "Max=118". This message means that the second parameter of the I command cannot exceed 118.

USING YOUR NEW RAMDISK

You are now ready to use your new ramdisk. Try copying some files to it. If one of these files is a binary load file, try loading it to RAM and see how fast it loads. Get a Directory listing:

♥B<RETURN>

to verify the ramdisk contents.

SELECTING YOUR RAMDISK AS THE SYSTEM DRIVE

You may want to select your ramdisk as the System Drive. (See p. 12, and also the S command, in your TOP-DOS Manual.) The command:

```
▼S S8<RETURN>
```

selects your ramdisk as the System Drive. If you choose to do this, be sure to copy the DUP.SYS file to your ramdisk, since TOP-DOS goes to the System Drive to load the DUP.SYS file when you go to DOS from BASIC or another user program.

USING A "HELLO" FILE TO SETUP YOUR RAMDISK

The HELLO facility of TOP-DOS may be used to initialize and set up your ramdisk on Boot-up. Just create a command file with the desired TOP-DOS commands, and name it HELLO. (See the Q command and HELLO facility descriptions in your TOP-DOS Manual.)

Your TOP-DOS disk contains a sample HELLO file, named HELLO.RAM. This file selects your ramdisk as the System Drive and leaves it containing the DUP.SYS file. The process consists of setting your ramdisk as the System Drive, initializing the ramdisk, writing the DOS files to it, deleting the DOS.SYS file from it, and finally getting a Directory listing.

NOTE: This method of writing DOS files and then deleting DOS.SYS to leave DUP.SYS is much faster than copying DUP.SYS from a real disk drive, as you can see by trying both methods.

To activate this HELLO.RAM file, copy it, as HELLO, to the disk that you wrote the augmented TOP-DOS to:

```
▼C D8:HELLO.RAM,D2:HELLO<RETURN>
```

Or, you can execute this file directly, using the Q command:

```
▼Q DI:HELLO.RAM<RETURN>
```

CREATING YOUR OWN HELLO FILE

You will want to create your own HELLO file. If you don't want your ramdisk to be your System Drive, you will not need to have the DUP.SYS file on it. But you will probably want to copy some of your most frequently used programs to it, so that you can access them quickly.

CUSTOMIZING YOUR NEW RAMDISK

You may modify your new ramdisk to suit your particular needs. This can be accomplished by including the commands in the HELLO file, or you can make the changes when you want.

The following material is primarily for the ADVANCED USER and assumes familiarity with the characteristics of your ramdisk.

SETTING THE RAMDISK SIZE

You can initialize your ramdisk to a different size than the default size which is used when you don't give a second parameter to the I command. For the AXLON ramdisk, you may use 98 (720/8) as the second parameter to make the ramdisk look like a standard ATARI drive. This will allow you to use the J Duplicate Disk command with your ramdisk.

If you have the MOSAIC SELECT RAM, you must have a total of at least 144K of RAM to be able to accomplish this same result. That is, you will need at least two 64K SELECT boards to get a ramdisk which is the full size of an ATARI drive.

Ramdisk Size Limits

If you give the second parameter of the I command, it must be at least 46 to accommodate the VTOC and File Directory. Furthermore, you must be careful not to specify a parameter which makes the ramdisk appear to be larger than the memory which is available to the ramdisk.

For the AXLON ramdisk, the available RAM is 128K, minus 16K which is needed for program memory. Thus, 112K of RAM is available for the AXLON ramdisk. This translates to a maximum value of 112 for the second parameter of the I command.

For the MOSAIC ramdisk, determine the total RAM in your system. Then subtract 48K for program RAM to get the RAM available for the ramdisk. Use this figure as the maximum for the second parameter of the I command.

NOTE: The the second parameter of the I command is the same as the number of "K" of RAM available for your ramdisk. This coincidence is a result of the fact that there are eight sectors per "K" for single-density drives, and the second parameter of the I command is the number of sectors divided by eight.

CHANGING THE DENSITY OF THE RAMDISK

If you have enough RAM, you can change the density of your ramdisk to double-density if you wish. Just use the S command.

The sector size for double-density drives is 256 bytes, which is twice the size for single-density drives. Therefore, you must have twice as much RAM to meet the minimum necessary to contain the UTOC and File Directory. This minimum for double-density is 96K. The AXLON has 112K available for the ramdisk, unless you choose to use some of its banks for other purposes (see below).

For the MOSAIC, you need at least 48K for program memory, so the minimum is $96+48=144$ K of total system RAM for double density.

REMOVING THE RAMDISK FROM THE DRIVE LIST

If you wish, you can remove the ramdisk from the drive list, using the R parameter of the S command. This will eliminate the need for a drive buffer for the ramdisk and thus reduce MEMLO when you are not using your ramdisk.

CHANGING THE DRIVE NUMBER

If you wish, you can change the drive number of your ramdisk. The drive number is stored at 1518 (hex). You can set it to any number from one to eight. Use the R command.

CHANGING THE BANKS

You can change the banks used for the ramdisk. There are three bytes which control the bank selection:

- 1A80 - Standard bank number for program memory
- 1A81 - First bank number for ramdisk
- 1A82 - Next bank number for ramdisk

The addresses are in hex.

Standard Bank Number

The "standard" bank number is that which is used for program memory. For the AXLON ramdisk, it is the bank used for memory at 4000-7FFF. For the MOSAIC, it is the bank at C000-CFFF.

The default for AXLON is bank #0. For the MOSAIC, the default is "no-bank", which is represented by the value FF.

For the MOSAIC, if you change the standard bank number from FF to a real value, you will extend the RAM available to programs from 48K to 52K. However, this puts Screen RAM in a MOSAIC bank. The result will be a flashing and tearing of the Screen display during any ramdisk activity. No harm results from this effect, but it could be annoying. Therefore, this configuration for the MOSAIC is not recommended.

First and Next Bank Numbers

The first and next bank numbers control the range of banks used for your ramdisk. By the "next" bank number is meant the last bank number, plus one.

The default values for these bank numbers were selected to make use of all available RAM for your ramdisk. For the AXLON, the values are (1,8). For the MOSAIC, they are (0,n), where "n" is the total number of banks available for ramdisk use.

You may wish to use some of the banks for purposes other than for your ramdisk.

Default Bank Numbers

Bank	Address	AXLON	MOSAIC
Standard Bank	1A80	0	FF
First Bank	1A81	1	0
Next Bank	1A82	8	n

Where "n" is the number of banks available for ramdisk use.

Obviously, care must be used in changing any of these values. If you do change them, you should save the DOS files (H command) and re-boot -- or at least press SYSTEM-RESET.

Good Luck!



1058 MARIGOLD COURT
SUNNYVALE, CALIFORNIA 94086
(408) 246-8325