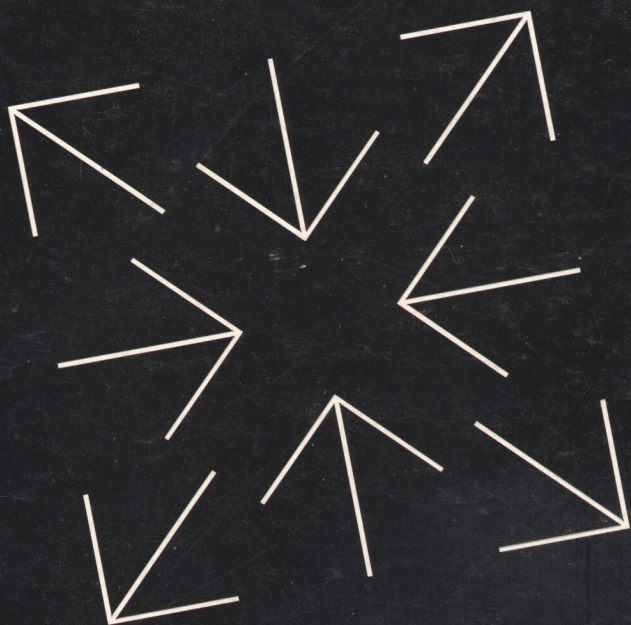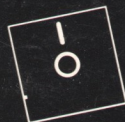# ATARI Disk Operating System Reference Manual

DOS 3

# ATARI DOS 3 REFERENCE MANUAL ERRATA
## 05/01/84

## MANUAL CORRECTIONS

PAGE 14, In the 'INTEREST' program, line 200 reads:
        200 LET A=P*(1+R/100) N
Change to:
        200 LET A=INT(P*(1+R/100)∧N+0.5)

PAGE 58, 'code 13', reads:
        13 = input and Output operation; positions file pointer to EOF; writing past EOF is allowed.

This line should be:
        13 = input and Output operation; positions file pointer to BEGINNING of file; writing past EOF is allowed.

PAGE 60, gives information on Random Access methods. The program indicates that only the 'X' parameter is significant, actually both the 'X' and 'Y' parameters are significant. For a correct demonstration of the Random Access NOTE and POINT commands contact ATARI Customer Relations at the address listed below.

PAGES 65 and 66, the 'BIRTHDAY' program requires several changes:

Line 140 reads:
        140 IF MONTH 1 OR MONTH 12 THEN?ERR$:GOTO 100
Change to:
        140 IF MONTH < 1 OR MONTH >12 THEN ?ERR$:GOTO 100

Line 240 reads:
        240 NEXT
Change to:
        240 NEXT I

Line 310 reads:
        310 IF A$ > < "YES" THEN GOTO 20
Change to:
        310 IF A$ < >"YES" THEN GOTO 20

Line 340 reads:
        340 ? "PLEASE TYPE PERSONS BIRTHDAY (MM-DO-YY)"
Change to:
        340 ? "PLEASE TYPE PERSONS BIRTHDAY (MM-DD-YY)"

Line 390 reads:
        390 TRAP 400:XIO 36,#3,0,0, FILE$:OPEN #,9,0,FILE$:GOTO 410
Change to:
        390 TRAP 400:XIO 36,#2,0,0, FILE$:OPEN #2,9,0,FILE$:GOTO 410

Line 620 reads:
        620 IF MONTH 12 THEN MONTH=1
Change to:
        620 IF MONTH >12 THEN MONTH=1

Line 700 reads:

        700 EOF=O;GOTO 600

Change to:

        700 EOF=O:GOTO 600

Line 3000 should be omitted.


# ATARI DOS 3 ANOMALIES

Early versions of ATARI DOS 3 used a random access method that was incompatible with large files. To determine if you have an early version, boot your copy of ATARI DOS 3 with ATARI BASIC, and execute the following command from ATARI BASIC:

# PRINT PEEK (1816)

If the value returned is '53', your copy of ATARI DOS 3 is a current version and does not require modification.

If the value returned is '51' or '56', please contact ATARI Customer Relations at one of the toll free numbers or address listed below, to obtain a program which will update ATARI DOS 3 to the current revision level.

# HOW TO USE THIS MANUAL

This manual has been developed with the user in mind. Each section of information represents one phase of the third version of the ATARI Disk Operating System (DOS 3). The newcomer to DOS can easily find the information needed to get started with DOS 3 without feeling encumbered by extraneous information. The experienced user, however, can quickly find and use the data required to perform more complex operations.

## FOR THE NEW DOS USER

Section 1 explains how to use this manual and the procedures for the most elementary operations:

- Definition of DOS
- Setting up the system
- Explaining the DOS Menu

Sections 2 and 3 discuss and explain everything about your diskettes from formatting to storage. The novice to DOS should read through Sections 1 and 2 before sitting down to work. This gives you an opportunity to become familiar with what you are to do.

## FOR THE NOVICE AND MORE EXPERIENCED USER

Section 3 starts your involvement with DOS 3, explaining how to identify your diskette files using filenames and filename extenders; this section also introduces you to Loading and Saving programs.

Section 4 is the crossover point between the new and more experienced user, and has a detailed description of each DOS Menu option and how to use it. Some of these options are only of interest to users familiar with the Assembler Editor cartridge and the hexadecimal number system.

## FOR THE MORE EXPERIENCED USER

Section 5 reviews the BASIC commands used with DOS 3 and gives sample programs showing the I/O commands in actual use. Each command format is also followed by an example showing the types of data going into each parameter. Section 6 contains further information about ATARI Disk Drives and diskettes (which may be of interest to both levels of users), along with details about storing and retrieving data. The balance of the manual contains a glossary of terms and additional information for the advanced user such as:

- Memory Maps
- Errors
- Saving RAM space

# TABLE OF CONTENTS

# 5 MORE USER INFORMATION

# 6 ADDITIONAL INFORMATION ABOUT THE DISK DRIVE SYSTEM

# APPENDICES

# INDEX

# ERROR SUMMARY

# GETTING STARTED WITH DOS 3 1

DOS (pronounced doss) is an acronym for Disk Operating System. Without a DOS, your ATARI Home Computer System cannot communicate with your ATARI 810™ or 1050™ Disk Drive. The DOS consists of comprehensive utility routines that allow you to:

- Store programs on diskette
- Retrieve programs from diskette
- Create and add to data files needed by programs
- Make copies of disk files
- Delete old files from a diskette
- Load and save binary files (for the advanced user)
- Move files to and from memory, the screen, diskette, and printer.
- Get help if you need it.

If you have never used a DOS before, you will find the ATARI DOS 3 simple to understand and easy to use. If you previously used the ATARI DOS 2 you will find differences in loading DOS 3 and changes to both the menu options and their command parameters. These changes make DOS 3 more advantageous because you have more available user memory space and greater flexibility than with the 9/24/79 version of DOS 1 or DOS 2.

## SETTING UP

To start, you must attach your ATARI 810 Disk Drive or ATARI 1050 Disk Drive to your ATARI Home Computer System, making sure your computer has at least 16K Random Access Memory (RAM). For complete instructions on setting up your equipment, please refer to the operator's manual shipped with your system. The procedures for attaching the ATARI 1050 Disk Drive and the ATARI 810 Disk Drive to your ATARI Home Computer System are contained in the respective Disk Drive Operator's Manual.

# ADDING MORE DISK DRIVES

Should you want to attach additional ATARI Disk Drives to your home computer system, you do so by "daisy-chaining." In the back of each ATARI Disk Drive are two outlets (labeled I/O CONNECTORS); attach the I/O cord from your second disk drive to Drive 1. Then attach the other cord from Drive 1 to the computer console.
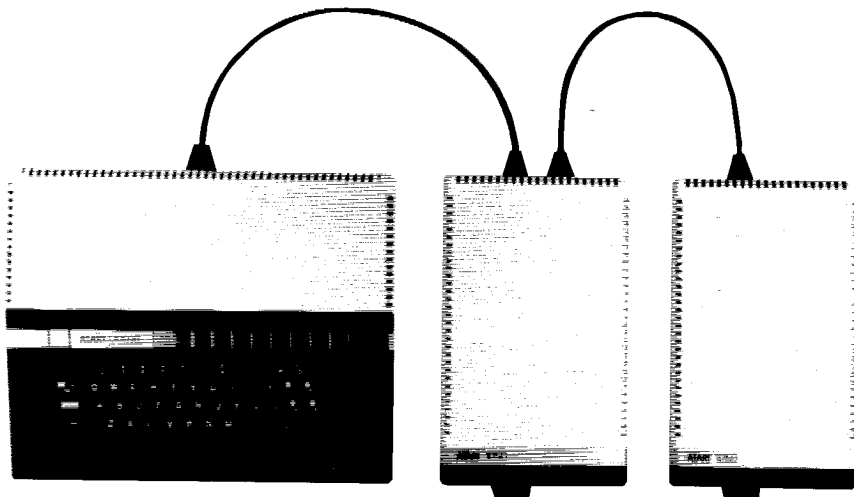
**Figure 1-1 Disk Drive Configuration**

# SETTING DRIVE CODES

Looking at the back of your ATARI Disk Drive(s), you can see a hole with two tabs. The white and black tabs must be moved on each drive according to that drive's designation. Refer to Figure 1-2 for the proper drive code setting.
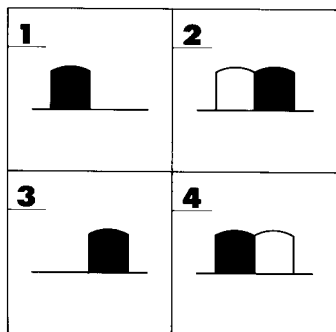
Using a pen or a small screwdriver, set the switches in the window to match the patterns shown here for Drive 1, Drive 2, and so forth. You must always have one drive set as Drive 1.

**Figure 1-2. Drive Code Settings**

2

# LABELING YOUR DISK DRIVES

Once you have properly set the drive codes, label each disk drive with its appro-
priate number so you do not make a mistake when using disk drive functions. It is
imperative that your Master or System Diskette ALWAYS be placed in Disk Drive 1.

# INSERTING A DISKETTE

Inserting a diskette into an ATARI Disk Drive is a simple, but very important pro-
cedure. If the diskette is improperly positioned, it can cause boot (starting-up) er-
rors during DOS loading procedures, and can also damage the diskette.

   Turn on the disk drive(s), and wait for the BUSY light to go off. Insert the diskette
as follows:
1. Remove the diskette from its protective paper sleeve.
   **Caution:** Hold the diskette ONLY by its black, sealed envelope. DO NOT touch
   any exposed surfaces of the diskette, as this can impair or destroy its read/write
   capabilities. DO NOT hold the diskette by placing your fingers through the
   center hole. DO NOT try to remove the diskette from its black, sealed envelope.
2. Hold the diskette so the labeled side is up, with the label toward you, and the ar-
   row on the label is pointing toward the disk drive door (see Figure 1-3). Also, if
   your diskette has a write-protect notch, this should be on your left. Please note
   that your Master Diskette does not have a write-protect notch because it is
   automatically write-protected at the factory.
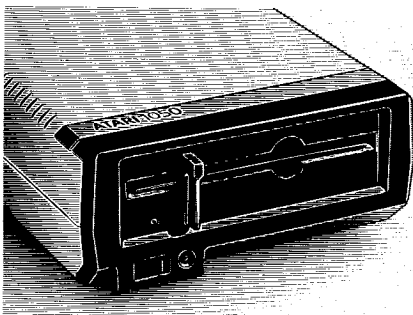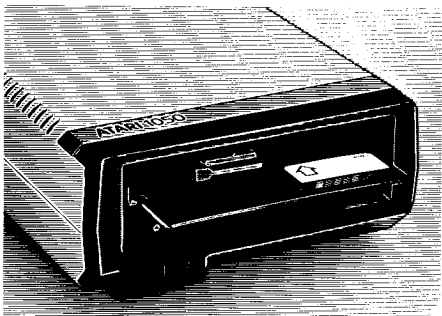


**Figure 1-3. Inserting a Diskette Into a Disk Drive**

3. Open the door to the disk drive and gently but firmly slide in the diskette.
4. Close the disk drive door.

3

# WHAT IS A DOS MENU?

The DOS Menu is loaded into your computer from your Master Diskette, which is always inserted into Disk Drive 1. The DOS Menu is similar to a restaurant menu; a selection of applications is presented to you on your television screen. You make your selection by typing the appropriate code letter and pressing **RETURN**. Your selected application is now available for use.

# HOW DO YOU CALL UP A DOS MENU?

After you have your television set turned on, your Master Diskette inserted into the disk drive, and your disk drive turned on, you are ready to load DOS 3 into the computer memory as follows:

### WITH A PROGRAMMING LANGUAGE CARTRIDGE INSTALLED OR WITH BUILT-IN BASIC
When you turn your computer on:
1. The BUSY light on the disk drive goes on during the loading process. DO NOT attempt to remove the diskette while this light is on. If you have inserted the ATARI BASIC cartridge or if your ATARI Computer has built-in BASIC , a **READY** prompt appears on the screen when DOS 3 is loaded. (If you have inserted the Assembler Editor cartridge, the prompt is **EDIT**.) This completes the first part of the loading procedure.
2. Type DOS and press **RETURN**. The DOS 3 menu displays on the screen (Figure 1-4). This completes the second part of the loading procedure.
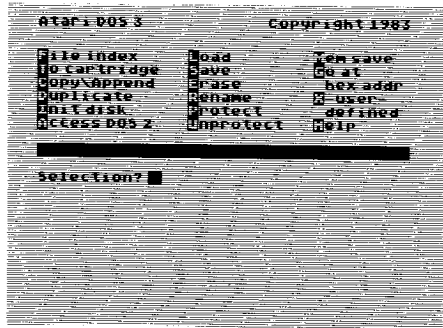


**Figure 1-4. The DOS 3 Menu**

With a cartridge present, the first part of the load brings in the FILE MANAGE-MENT SYSTEM (FMS) (from file FMS.SYS) and KEYBOARD COMMAND PRO-CESSOR (KCP) resident (from file KCP.SYS). The second part brings in the KCP overlay (from file KCPOVER.SYS), which displays the menu and process commands.

4

## WITH NO CARTRIDGE INSTALLED

When you turn your computer on, DOS 3 loads entirely without user interaction. The BUSY light on the disk drive goes on during the loading process. DO NOT attempt to remove the diskette while this light is on. The DOS menu is displayed automatically when DOS 3 is loaded.

After you make a code letter selection, a prompt message displays on the screen requesting information from you. This capability makes it a "self-prompting" menu. The prompt message that appears most frequently reads:
SELECTION?

This means the system waits for you to do one of the following:

- Type in one of the letters or
- Press **ESC**, which redisplays the DOS menu.

# WHAT YOU SHOULD KNOW BEFORE USING DOS 3

# 2

This section provides a set of definitions of terms used often in the descriptive sections that follow. As specified in an earlier chapter, these terms are repeated in the glossary. It is important that these specific terms be somewhat familiar to you so that you can better understand how DOS 3 functions. Here are the definitions:

## FORMAT

To format (or to initialize) a disk means to prepare a disk for use. When you purchase a new blank disk, it has no data written on it. DOS 3 must be told to use the INIT (initialize) function selection of the main menu to format the disk.

The INIT function uses a precise set of mechanical steps to stop at a number of different positions on this blank diskette. Each of these positions is defined as a track. While at each of these positions, it writes data on the magnetic surface to separate each track into "sectors."

A sector is an individually-accessible piece of the track. Each sector can be located later by the file management system (known as FMS.SYS on your ATARI DOS 3 Master Diskette.) The FMS organizes these sectors into a set of "blocks" used to define the amount of space available and used on the diskette. Blocks are specified in the section on the File Index Option.

Because this information is magnetically recorded, exposing the disk to any kind of magnetic field may destroy your data. It is also important, as mentioned earlier, to protect the recording surface from scratches or dirt that would make it more difficult for the disk unit to "read."

For more information about how to format a disk, see the section titled "How To Use The INIT Function."

## FILESPEC

Filespec is a shorthand way of saying "file-specification." DOS 3 uses the name Filespec any time it wants you to specify the device, device number, and the name of the file you want DOS 3 to use. A "file" is a grouping of data that is related in some manner.

## DEFAULT SPECIFICATIONS

ATARI DOS 3 has a number of so-called "Default" specifications. This means that if you don't choose any specific item, ATARI DOS 3 chooses an appropriate item "by default."

In most cases, when a command question appears, there is nothing else on the command line except the question itself.

If the default selection has not appeared yet on the command line, and if you might use the default selection, press **RETURN** once. This makes the default selection appear. If you wish to change the default, use your command editing

keys as noted in the section titled "Command Editing" to move the cursor and change the response before again hitting **RETURN**. Note that for some commands, the default selection appears immediately along with the question. Therefore, look at the screen before hitting **RETURN** each time. (You may wish to make changes.)

Instead of accepting the default response, you may type your own response on the command line. Once there is a response on the command line, press **RETURN** for ATARI DOS 3 to accept that response.

If you make a mistake in the command line before you press **RETURN**, you can use the command edit keys to make a correction. After you hit **RETURN**, you have to either **ESC** to the main menu again, or preferably just restart the current command using the **SHIFT/CLEAR** or **CTR/CLEAR** key grouping. See the section titled "Command Editing" for more information.

# WILDCARDS

ATARI DOS 3 recognizes two "wild-cards" that you can substitute for characters in a filename. Wild cards are represented by the special characters question mark (?) and asterisk (*).

When DOS 3 encounters a wildcard character, it is told to find and use all filenames which match the "general description" of the filename you have specified this way. Each of the wildcards is treated as a "don't care" item within the file name. The question mark is the single-character wildcard. The asterisk is the multiple character wildcard.

For example:
**PROG?.BAS**

The wildcard character (?) indicates that you don't care what the fifth letter is, just use any file in which the rest of the letters match. In this example, the names PROG1.BAS, PROG2.BAS, and PROGZ.BAS match and those files would be used. A name like PROG12.BAS would not match because the "2" would not have matched the blank in the original filespec. This is shown in the sample with the filespecs spread out to match the allowable maximum width shown in the section titled "FILESPEC" above:

| | | |
|---|---|---|
| Wildcard example: | **PROG?** | **. BAS** |
| (8 chars max.) | –––––––– | ––– (3 chars max.) |
| Actual file on disk | **PROG12** | **. BAS** |

Notice that you do not need to specify the blank spaces. DOS 3 assumes that they are there if the left-hand part of the filename is less than 8 characters. In other words, the left hand side of the filename is "left-justified" in the field of 8 characters.

For example:
**PROG***

The wildcard character (*) indicates that the filename is a match for any name in the file index that starts with **PROG** and ends with anything else, such as:

**PROGRAM   PROGRESS   PROG12**

The filespecs above do NOT match any of those given in the previous examples (PROG1.BAS, etc) because there was no "extension" specified. To match any of the file names with extensions, you would have had to specify:

**PROG*.***

This combination means that any letters can follow the **PROG** part of the word, or any extension may be used (all blanks or anything else)."

8

The asterisk, as a wildcard character, can appear a maximum of once either on the left of the separating point (.) and once on the right. It always means "the rest of the characters, including the one at this position."

Examples of the use of wildcards are used throughout this manual, particularly in the section on the File Index function.

The following is a summary of the DOS menu options to show whether they allow you to use wild cards in their parameters.

| DOS MENU OPTION | WILD CARDS |
|---|---|
| A. Access DOS 2 | Yes |
| C. Copy File | Yes |
| D. Duplicate Disk | No |
| E. Erase File | Yes |
| F. File Index | Yes |
| G. Go At Hex Address | No |
| H. Help | No |
| I. Init Disk | No |
| L. Load | Yes |
| M. Mem. Save | No |
| P. Protect | Yes |
| R. Rename | Yes |
| S. Save | No |
| T. To Cartridge | No |
| U. Unprotect | Yes |
| X. User-Defined | No |

# GETTING HELP

There may come a time when you have accidentally entered the wrong information or the wrong function. Or you may simply be unsure about what to do next. At this point you may either decide to start another command or call the Help function.

DOS 3 provides two different ways to exit from a command.

The first way is the **ESC** (or escape) key. This immediately halts the current command if you are in the data entry screen for that command and returns you to the main DOS 3 menu.

If, instead of returning to the main menu, you would like to get some helpful information about the command, you can press the **inverse video (ATARI)** key or the **HELP** key. This performs slightly different functions depending on which of the functions you need Help with.

For the functions:
Init Disk,
Duplicate,
Copy/append, and
Access DOS 2,

the Help function is built into the utility routine itself. This means that you can ask for Help, then use the **ESC** key to return to the data entry screen again.

You may also switch back and forth as many times as you wish as you are entering the data by using the **inverse video** key or the HELP key to go to the Help screen for the function, and, as indicated on the Help screen itself, press the **ESC** key to return to the data entry screen. Each time you leave a function, anything you entered is lost. The **ESC** key, when pressed from the data entry screen, cancels the current command and returns you to the main menu.

For all commands other than those mentioned in the previous paragraph, pressing the **inverse video** key or the **HELP** key brings up a help screen for the function being performed. This function's help screen is called by loading the normal Help

function. This cancels the current command and places you in control of the Help function. To reenter the main menu, **ESC**ape from the Help function when you have read the desired screen, then reselect the original function.

# COMMAND EDITING

DOS 3 provides, for most of the command options, a default data entry if no user entry has been selected. The default selection, if not already present on the command line along with the command question, may be made to appear by pressing **RETURN** once. The keys listed in the following paragraphs result in the specific actions shown. If a key is pressed that the system does not recognize, a beep tone sounds to indicate a data error. Recheck your entry and try again. If you don't want to use the default option, you can change the command line using the following keys:

### DELETE/BACKSP

This key backspaces across each character, deleting as it goes. It allows you to retype whatever command you wish in its place.

### CTRL/RIGHT ARROW

This key combination moves the cursor to the right not deleting or otherwise changing the characters it passes through. This is useful for making a change to a single character on the command line.

### CTRL/LEFT ARROW

This key combination moves the cursor to the left not deleting or otherwise changing the characters it passes through. This is useful for changing a single character to a different character in the command line.

### SHIFT/DELETE

This key combination completely erases the current response on the command line, placing the cursor at the first space immediately following the command question.

### CTRL/CLEAR
### SHIFT/CLEAR

These key combinations completely erase all preceding sections of the command data entry and allow you to start again. A case where this might be used is if you have noticed an error in a previous data entry line. DOS 3 does not allow the cursor to be moved to a previous data entry for correction, but rather always keeps the cursor operating on the line where the question is located. **SHIFT/CLEAR** means "restart the command."

### ESCAPE KEY

This key allows you to escape entirely from the command entry mode and returns control to the main DOS 3 menu.

### INVERSE VIDEO KEY, HELP KEY

Either of these two keys, when used from the command line entry mode, temporarily exits this mode, and presents you with helpful information about the command. Note that in order to use this function, on the disk in drive 1, the following files must be present: HELP.UTL and HELP.TXT.

# IDENTIFYING YOUR DISKETTE FILES

Files are classified into two types:

**PROGRAM FILES.**

These are sets of instructions that tell the computer to perform specific tasks.

**DATA FILES.**

These usually contain the information used by a program file, but not the instructions. For instance, a permanent data file may be a name and address file capable of being updated at any time.

Just as you call a person by name, you must call a file by name when you want to access it. The filename on the diskette is part of the file specification (or filespec for short). Filespecs (Figure 2-1) have six key elements. If you call a file by its wrong name, just like a person, it won't answer; instead, an ERROR-170 appears on the television screen.

The rules for filenames are:
- The maximum length of a filename is eight characters.
- The only characters that can be used in a filename are the letters A through Z, and the numbers 0 through 9.
- The characters * and ? CANNOT be used as part of a name when establishing a filename. (See the section on Wildcards for explanation.)

# FILENAME EXTENDERS AND THEIR USE

You can add a three-character extender to a filename to indicate the type of data in a file. You can use any legal combination of letters and numbers, for example:

**SYS** system files
**BAS** BASIC program files
**DAT** data files
**MUS** ATARI Music Composer™ files
**ASM** assembly language files
**OBJ** binary load files
**SRC** source files
**LST** files created by the LIST command
**SAV** files created by the SAVE command

"D 1 : A T A R I 8 0 0 . B A S"

Device Code

Device Number (optional)

Required Colon

Filename (up to 8 characters)

Period required as separator if extender is used.

Extender (optional) Includes 0-3 characters
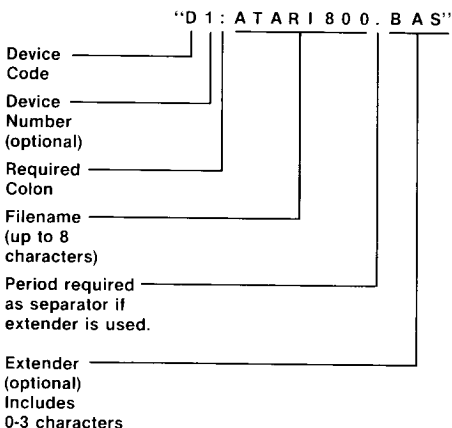
**Figure 2-1 Structure of a Filespec**

11

If you try to use an extender that has more than three characters, DOS 3 ignores the additional character(s). The following summary illustrates both legal and illegal filenames, with an explanation of what makes a name illegal:

**CASHFLOW**    Legal name.
**ATARI.BAS**    Legal name.
**3ATARI.DAT**    Legal name.
**ATARI22.ASM**    Legal name.
**ATARI#**    Illegal name.
**A1234567.BA2**    Legal name.
**B ATARI.LST**    Illegal name. No spaces allowed.
**FMS.SYS**    Reserved for DOS.
**FMSSYS**    Legal name.
**TEST1.123**    Legal name.

# BOOT ERRORS

When you start your system, boot errors can occur for the following reasons (Figure 2-2):
1. The inserted diskette does not have DOS on it.
2. The diskette was inserted wrong.
3. The diskette has been scratched, warped, or marred. In this case, use another diskette.
4. The diskette is a double-density diskette in an ATARI 810 Disk Drive.
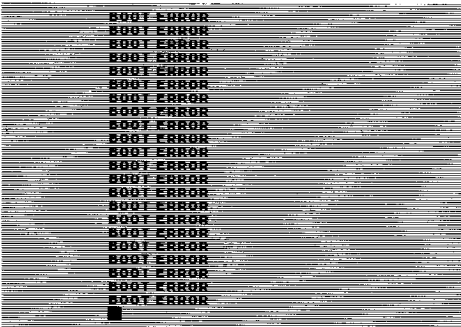


Figure 2-2 Boot Errors

The following conditions also cause a boot error, but no indication of it appears on the screen:
1. The disk drive was turned on AFTER the computer console was turned on.
2. The disk drive is not properly connected to the computer console.
3. The power adapter plug has loosened from its wall socket.
4. The power adapter plug has loosened from the disk drive PWR socket.
5. The drive code setting is not correct.

If you have checked and find none of these problems, take the following steps:
1. Insert the Master Diskette or a System Diskette into Drive 1 and reboot the system.
2. Remove the Master Diskette and store in a safe place.
3. Reinsert the problem diskette and save any accessible files on another diskette using the process for copying files (see the COPY/APPEND Menu option in Section 4).
4. Then with the problem diskette in Drive 1, use the ERASE function to erase all the files.
5. Try using the diskette again. If this fails, the diskette will have to be reformatted.
   **Note:** For the CX8111 ATARI 810 Formatted Diskette 3, this should be done ONLY AS A LAST RESORT to avoid losing the improved formatting.
6. If reformatting fails, the diskette has bad sectors on it and should be discarded.

# SAVING, LOADING, AND RUNNING PROGRAMS

After you have created your System Diskette and, if necessary, formatted a blank diskette, you are ready to write your own programs. When the power is turned OFF on your computer, you lose any program stored in memory. With an ATARI Disk Drive, you have the means to store and retrieve programs without having to retype them. You can use the following simple BASIC commands to store and retrieve your programs. We have included a sample program for you to type into your computer and a step-by-step procedure for saving it on a diskette and loading it back into the computer.

If You Have an ATARI 810 Disk Drive:
1. Turn on the disk drive.
2. Insert the System Diskette in Drive 1.
3. Turn on the computer console and television set.
4. When the BUSY light goes off, remove the System Diskette, and insert a diskette that has been formatted.
5. Type the program shown in the following example.
6. Type SAVE "D:INTEREST.SAV" **RETURN.**
7. When the BUSY light goes off and the READY prompt message appears on the screen, the program you typed in is successfully saved on the diskette.
8. Type NEW/RETURN to erase the program from RAM.

   Now to reload the program into memory:
9. Type LOAD "D:INTEREST.SAV"/**RETURN.**
10. When the READY prompt appears on the screen, you can now run the program by typing RUN/**RETURN.**
11. You can also load and run your program by typing RUN "D:INTEREST.SAV"/**RETURN.**

**Note:** Do not delete the Sample Interest program from your diskette; you will use it again in Section 5.

13

```
100 REM *** INTEREST
110 PRINT"IF YOU TYPE THE AMOUNT OF P
RINCIPAL"
120 PRINT"AND THE INTEREST RATE PER Y
EAR, I WILL"
130 PRINT"SHOW YOU HOW YOUR MONEY GRO
WS, YEAR BY"
140 PRINT"YEAR. TO STOP ME, PRESS THE
BREAK KEY."
150 PRINT
160 PRINT"PRINCIPAL";
165 INPUT P
170 PRINT"INTEREST RATE";
175 INPUT R
180 LET N = 1
190 PRINT
200 LET A = P*(1 + R/100) N
210 PRINT"YEAR = ";N
220 PRINT"AMOUNT = ";A
230 LET N = N + 1
240 GO TO 190
READY
```

# THE DOS 3 MENU SELECTIONS

The following paragraphs explain the various selections you can make from the
DOS 3 Main Menu. In each paragraph, there is a brief explanation of what each of
the functions does.

Please do not try to use these options until you understand them thoroughly.
There is a separate section in this manual to explain each of them.

### File Index

Selecting this item allows you to view an index of the files on a diskette or simply
to find out if a file of a particular name is on the disk with which you are working.
See the section titled "How to use the File Index Function" for a detailed
description.

### To Cartridge

This selection directs DOS 3 to give control to a cartridge you have installed in the
left-hand slot of your ATARI 800™ or in the cartridge slot of your ATARI 400™ or
1200XL™. For more information about what happens after this selection is chosen,
see the section titled "How to use the To-Cartridge Function."

### Copy/Append

Selecting this function allows you to make copies of your program or data files.
The copies may be made using either a one or a multiple disk system, or, if it is a
"text" file, the copy may be directed to your television screen or your printer. See
the section titled "How to use the Copy/Append Function" for more information.

### Duplicate

Selecting this function allows you to make a complete copy of an unprotected
ATARI program disk. It includes the capability to initialize a disk (see below for Init
option). It performs a block-by-block copy from one disk to another, using either a
single-disk or a multi-disk system. For more information, see the section titled
"How to use the Duplicate Function."

### Init Disk

Selecting this function allows you to initialize, or prepare a disk for use. It is also known by the name "Format" because it writes a certain pattern on a disk that allows it to read and write data onto it later. For more details about using this option, as well as a description of what Format really means, see the section titled "How to use the Init Function."

### Access DOS 2

For those of you who have purchased or developed software written to run under ATARI DOS 2, this function allows you to use your disk drive to read and convert such programs and data so they run under DOS 3. For complete information showing how to run the conversion, see the section titled "How to use the Access DOS 2 Function."

### Load

This function allows you to put back into main memory any data you have saved with the Save function. For more details, see the section titled "How to use the Load Function."

### Save

This function allows you to save the contents of specified memory locations on disk. It is normally used with machine language programs. See the section titled "How to use the Save Function" for more details.

### Erase

This option allows you to delete one or more files from a diskette. It is normally used to make room for data on the diskette when that old data is no longer needed. Once a file has been erased, its data no longer can be recovered. For more information, see the section titled "How to use the Erase Function."

### Rename

This function allows you to change the name of a file currently in the File Index of a diskette. It is often used to simply make a name more meaningful than that used to put it on the disk in the first place. More information about this function is shown in the section titled "How to use the Rename Function."

### Protect

This function allows you to tell DOS 3 that a particular file should be protected. This means that DOS 3 should mark the file name in a special way to prevent it from being erased accidently. In fact, if you try to erase a protected file using the DOS 3 Erase command, it reports an error. For more information about this option, see the section titled "How to use the Protect Function."

### Unprotect

This function is the reverse of the one above. It removes the protection from a specified file so that it can be erased without causing an error report. See the section titled "How to use the Unprotect Function."

### Mem.Save

This function allows you to create a file on the diskette to store the contents of memory while part of DOS 3 is being used. It is often used by the advanced user who goes from a programming language to DOS and back often during program development. It is used to save the program being developed in memory when DOS 3 is active. It is restored to that memory area DOS 3 had temporarily used when the user returns to the programming language. For detailed information on Mem.Sav, see the section titled "How to use the Mem.Sav Function."

### Go at Hex Addr

This function is used by the machine language user to start the execution of a pro-
gram that has already been placed into the memory system. It requires knowledge
of machine language programming. For more information, see the section titled
"How to use the Go-at-Hex-Addr Function."

### X-User-Defined

This selection allows you to define the name of a machine language program, sav-
ed on the diskette using the Save option, to load and run. For details about this
selection, see the section titled "How to use the X-User-Defined Function."

### Help

This option literally can Help you learn about the operation of DOS 3. All of the in-
formation you need about how to operate DOS 3 is contained in this manual. If you
need a quick reminder about some of the possible option choices, the Help func-
tion may be used to get that information. For more data on the Help Function, see
the section titled "How to Get Help." Note that in some of the selections, it is
possible to obtain Help while you are entering data for the control of the function.
These details are specified in the section referenced above.

## WHAT ELSE IS ON THE MAIN MENU?

There is a solid white line shown on the main DOS 3 menu. This is the error or
results reporting line. It normally remains a solid color. When errors occur, this line
specifies the error number and a message which should help you determine what
to do next. During certain functions, such as protect and unprotect, this line is also
used to report function progress.

A list of the possible errors and some suggestions for probable causes are con-
tained in a separate appendix E, titled DOS 3 ERROR MESSAGES.

The final item on this main menu is "Selection?". DOS 3 is asking you to select
an option from the menu. Before you select, make certain you are already familiar
with the terms shown in the section titled " What You Should Know Before Using
DOS 3." Then, for each option, turn to the section where that option is explained to
show you exactly how it is to be used.

# DISKETTES 3

## THE MASTER DISKETTE

The Master Diskette contains the disk operating system programs. These programs include all the system file management and utility routines necessary to make your disk drive function with your ATARI Home Computer System. Without a disk operating system, you cannot access the disk drive.

Each Master Diskette contains the following files:

### FMS.SYS File

FMS.SYS is a file containing the File Management System. FMS.SYS contains the subfunctions that are provided by the FMS: ERASE FILE, RENAME FILE, PROTECT FILE, UNPROTECT FILE, and LOAD.

### KCP.SYS File, KCPOVER.SYS File

This combination contains the DOS Menu and DOS subfunctions NOT provided by the FMS. These files make up the Keyboard Command Processor (KCP). Whenever you want to see the DOS Menu or perform these DOS subfunctions (SAVE, GO AT HEX ADDRESS, TO CARTRIDGE, COPY FILE, INIT DISK, DUPLICATE DISK and ACCESS DOS 2), you must load the KCPOVER.SYS file into RAM by typing DOS and pressing RETURN.

**Note:** Normally when you bring the KCPOVER.SYS file into RAM, it writes over data in the lower program area occupied by BASIC or assembly language programs. However, when you create a MEM.SAV file (see section on MEM.SAV) on your diskette, DOS saves any data in RAM to diskette before loading KCPOVER.SYS When you are finished using the KCP functions, MEM.SAV allows you to reload your program automatically.

### HANDLERS.SYS File

This file is used to poll (check) the peripheral units (if any) attached to your ATARI Home Computer System and to run machine code programs (see Section 4 of this manual for more detail on AUTORUN.SYS). If this file is on the drive 1 diskette during boot up, it is loaded and run automatically.

**HELP.UTL**—The HELP utility

**HELP.TXT**—Text information displayed by the HELP utility

**INIT.UTL**—The INIT Disk utility

**CONVERT.UTL**—The ACCESS DOS 2 utility

**DUPDISK.UTL**—The DUPLICATE utility

**COPY.UTL**—The COPY/APPEND utility

# ATARI FORMATTED DISKETTES II

In addition to the Master Diskette 3 (CX8104), your ATARI Disk Drive comes with an ATARI Formatted Diskette 3 (CX8111). Although this diskette has no files or program data on it, it has been preformatted at the factory. Preformatting means the diskette was divided into tracks and sectors before packaging (Figure 3-1) so that it has an improved sector layout. This improved sector layout makes it possible for you to store and retrieve information more rapidly than is possible with diskettes formatted on your ATARI Disk Drive. This"empty"diskette is provided so you can make a backup copy of your Master Diskette. This backup copy, called a System Diskette, is the one you actually work with, ensuring the safety of your original Master Diskette (see Making a System Diskette).

**Formatted Diskette**



Sector (section of track)

Track (one revolution)

Write-Protect Notch

Timing Hole

Read-Write Area
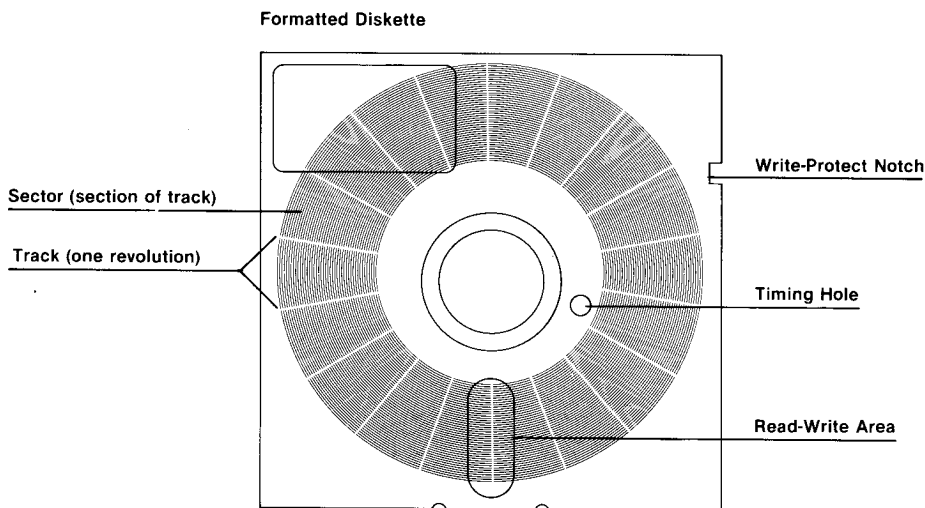
**Figure 3-1 A Formatted Diskette**

If, at any time, you decide to erase the files from a preformatted diskette, use the DOS Menu selection **E**, ERASE FILES. This saves the improved sector layout that allows the diskette to run faster. Note that if you reformat the diskette for any reason (perhaps to test for possible bad sectors), it no longer has the improved sector layout.

**Note:** You can also use blank diskettes that have not been preformatted at the factory with the ATARI Disk Drive. You can format these diskettes yourself using the INIT menu option before duplicate disk or store programs on them (see How to Format a Diskette). However, you do not have the advantage of the improved speed of the preformatted diskettes.

# HOW TO FORMAT A DISKETTE

You need to format any blank diskette before you can write on it. Formatting organizes a diskette so DOS 3 knows where information is located. Unlike a phonograph record that has visible spiraling grooves imprinted onto it, the diskette has magnetically encoded tracks that are not visible.

Once you have formatted a diskette, it contains 40 concentric tracks, which are divided into 18 (26 for double density) pie-shaped wedges called sectors. You ascertain the storage capacity by multiplying the number of tracks (40) by the number of sectors per track (18), which gives you 720 sectors. Each sector can store 128 bytes of data. These sectors are, in turn, organized by the FMS into blocks of 8 sectors each. Therefore a single-density, freshly formatted disk contains 720/8 = 90 blocks. A double-density formatted diskette for the 1050 contains 1040 sectors or 130 blocks.

Three blocks are used for booting the system and for the file index. As a result, you actually have a total of 87 blocks to which you may write data. The ATARI 1050, in double-density mode, has 127 blocks available.

To format a diskette, you must use the I, INIT DISK option on the DOS menu. If you are using an ATARI 810 Disk Drive as Drive 1, see instructions below. If you are using an ATARI 1050 Disk Drive as Drive 1, the same instructions are valid because you are using the 810 single-density format.

**USING THE ATARI 810 DISK DRIVE:**
1. Turn the disk drive on. Wait for the **BUSY** light to go off.
2. Make sure the switch on the back of the disk drive is set to No. 1 (refer to Figure 1-2 for drive code settings).
3. Insert the ATARI 810 Master Diskette and close the drive door.
4. Turn the computer console on. DOS loads into the computer memory.
5. When the **READY** prompt appears (if you have inserted the ATARI BASIC cartridge or your computer has built-in BASIC), type DOS and press **RETURN**. After a few seconds the DOS Menu appears on the screen. (If no cartridge is inserted, the DOS Menu appears on the screen automatically.)
6. Type I for the INIT DISK option and press **RETURN**.
7. When the prompt message **Format disk in drive (1-8)?** appears, remove the Master Diskette from the disk drive and insert a BLANK diskette. Close the door, and press **RETURN** twice. The first time you press **RETURN**, a 1 appears. This is called the default drive selection. The second time you press **RETURN**, you indicate to DOS 3 that drive 1 is to be used. CAUTION - NEVER FORMAT YOUR MASTER DISKETTE!
8. The next prompt says:
   **Format type?**
   **1 for single density**
   **2 for double density ?**
   Press **RETURN** twice. The first **RETURN** results in a 1; the second **RETURN** indicates to the system that you intend to use drive 1, the default drive.
9. The next prompt says:
   **Write FMS.SYS file#(Y/N)?**
   Type Y **RETURN**, then press **RETURN** again.
   This sequence indicates to DOS 3 that you are making a system diskette.
10. The final prompt used for this example is:
    **Modify FMS parameters (Y/N)?**
    Press **RETURN** twice to indicate to DOS 3 that a standard file manager is required.

11. At this point DOS 3 displays the parameters selected for this function, next it displays:
    **Press SHIFT-CLEAR to select different initial values or ...**
    **Insert diskette in drive #1 and press RETURN to initialize.**
12. CAUTION - FORMAT ERASES ALL DATA ON THE DISK BEING FORMATTED. When you press **RETURN**, the BUSY light comes on and the system formats the diskette.
13. When DOS 3 has completed its task, it prints:
    **Initialization completed.**
    **Press RETURN to init next diskette.**
    **Press ESC to return to DOS.**
    The diskette is now ready to use. Follow any of the specified instructions to proceed.
    If you have two or more ATARI 810 Disk Drives, you can format a blank diskette on any drive. However, you must know the drive code setting of the drive you use, so you can respond to the prompt:
    **Format disk in drive(1-8)?**

## MAKING A SYSTEM DISKETTE

The first disk operation you should perform is to duplicate your Master Diskette. This provides a backup in case the diskette you are using is damaged. For convenience, the duplicate of your master diskette is referred to as the System Diskette (working copy), and is the one you would normally use to load DOS 3 into RAM.

## WRITE-PROTECTING YOUR DISKETTES

Write-protecting is simply a method of preventing you from inadvertently writing over valuable information you may not want to lose from a diskette.
    You may notice that the DOS 3 Master Diskette has no notch on the left side of the diskette jacket so it is impossible to write files on it; therefore, it is already write-protected. Blank and preformatted diskettes do have the notches on the left side of the diskette jacket enabling you to write to the diskette.
    Normally, you would not write-protect a System Diskette, as this defeats the purpose of the MEM.SAV file, preventing you from writing the RAM-resident program to the diskette when necessary. For this reason, you may wish to put the files you want to save on another diskette, which you can write-protect.

## HOW TO WRITE-PROTECT VALUABLE DISKETTES

A sheet of large file identification labels and a second sheet of small adhesive write-protect tabs are included in each box of ATARI diskettes.
    Write-protecting is accomplished by simply removing an adhesive write-protect tab from the sheet and folding it over the notch on the edge of the diskette (Figure 3-2).
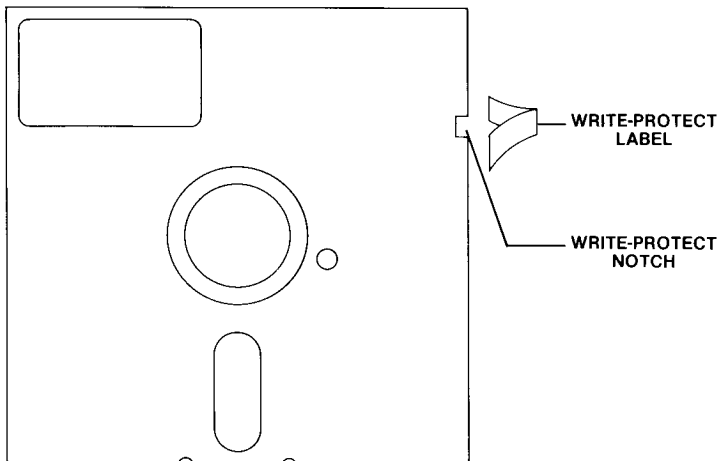
**Figure 3-2. Write-Protecting a Diskette**

If you try writing to a write-protected diskette, an ERROR-144 message is displayed on the screen.

# LABELING DISKETTES

Use the self-adhesive labels to identify the data on each diskette. Write on the label first to avoid damaging the diskette, and then attach these labels in the upper right corner of the diskette envelope.

# SINGLE-DENSITY AND DOUBLE-DENSITY DISKETTE RECORDING

The principal difference between the ATARI 810 Disk Drive and the ATARI 1050 Disk Drive is the way the data is encoded for storage on diskettes. Both the ATARI 810 and the 1050 drives store data on the diskettes in 128-byte sectors. In addition, the 1050 can read or write 810 compatible (720 sectors) diskettes or it can write in double-density mode (1040 sectors). Note however, that although the 1050 can read single-density 810 diskettes, the 810 cannot read double-density 1050 diskettes. Therefore, if compatibility is required between the 810 and the 1050, you should use the 1050 in single-density mode.

# HOW TO STORE DISKETTES

Since your diskettes are flexible, they are subject to damage. The following sugges-
tions are to keep your diskettes in good condition:

■ ALWAYS keep the diskettes in their protective paper sleeves when not in use.
■ Store them vertically as you would properly store records; do not stack them
  one on top of another.
■ Store the diskettes AT LEAST 12 inches from your television set or any other
  possible source of magnetic fields.
■ Store the diskettes away from any direct source of heat.

Your diskettes are an important and valuable part of your ATARI computer
system and, with proper care, should provide you many hours of dependable use
and enjoyment.

# SELECTING AND USING DOS 3 FUNCTIONS  4

## HOW THIS CHAPTER IS ORGANIZED

ATARI DOS 3 provides many features that should make it easy for the first-time user to get acquainted with and effectively use the system.

This manual begins with a section showing exactly how to get started. This is followed by a section on words the user should be familiar with before proceeding to read the rest of the manual. These terms are also included in the glossary at the end of the manual, however it is essential that those covered in this early chapter be known to allow the user to understand the other parts of the manual.

The rest of the manual is organized to allow an easy introduction to the first-time user. It is written as a semi-tutorial to ATARI DOS 3 in that for each command that DOS 3 can execute, the first thing encountered is always the most basic definition of the task that DOS 3 can perform. This is followed, in each case, by the exact, keystroke-for-keystroke command sequence that is needed to do the most common of the tasks that this command can perform.

Following the description of the most often used command sequence is the explanation of other features that are built into this command sequence. The first-time user may simply skip on to the next major section at this point each time, if one is simply trying to get a feel for "what is DOS 3."

## HOW TO USE THE ACCESS DOS 2 FUNCTION

This function is selected from the main DOS 3 menu by the letter A. It is used to convert existing files written for use with ATARI DOS 2 so that they can be used with DOS 3.

This conversion is necessary because the file structure of DOS 2 is different from that of DOS 3. DOS 3 cannot, without conversion, read the File Index or any of the files of a DOS 2 diskette. After the conversion is made, all files can be used normally with DOS 3 loaded.

On selection of the A option of the DOS 3 menu, the computer reports, on the DOS 3 menu screen, the following message:

**Loading D:CONVERT.UTL...**

This means that, to run this function, this program must be present on the diskette inserted into disk drive number 1. After the file has been loaded, the screen clears, and another screen appears. The options on the screen show the default selections. In other words, as each question appears on the screen, the RETURN key is pressed twice ... once to make the default selection appear, and the second time to accept the default.

The first question the computer asks is

**Source drive number?**

The default is 1. This is the DOS 2 diskette that you wish to read and converted to DOS 3 format.

The second question the computer asks is

**DOS 2 filename?**

The default is *.* which means ALL files on that disk. This utility can convert one, or only selected files. See the rules on wildcards in the section titled "What You Should Know Before Using DOS 3" for more information about file specifications.

The third question is

**Destination drive number?**

The default is 1 (a single drive conversion is ok). The files that match the source filespec are read from the source drive in DOS 2 format, and rewritten to the destination drive using DOS 3 format. If you have a two-drive system, it is most efficient to specify one drive as the source and the other as the destination. Otherwise, it is necessary to swap diskettes in and out of the single drive until the function is completed.

The fourth question is

**DOS 3 filename?**

The default is *.* which means make the file names the same on the DOS 3 destination disk as they were on the DOS 2 source disk. The filename contents (8 letters, plus 3 letter extensions) are the same from DOS 2 to DOS 3, so no meaning is lost in the translation.

After the questions have been answered, and you have pressed **RETURN** to enter the final response, the screen indicates either:

**Insert source disk, Press RETURN**

(if a single disk drive is being used)

or:

**Insert source disk,**
**Insert destination disk. Press RETURN**

(if two disk units are specified).

If a single drive is used, the prompt alternates between:

**Insert source disk**

and

**Insert destination disk**

Continue alternating until the conversion process is complete.

To avoid any possible errors be certain that the source diskette has been write-protected before you start.

**GETTING HELP**

This function has a built-in Help feature. You can ask for help at any time during the command entry phase by pressing the **inverse video (ATARI)** key or the **HELP** key from the command entry screen. When you have read the Help screens and wish to return to the data entry screen, press the **ESC** key. You may go back and forth to this Help screen as often as you like during the command entry phase by pressing **inverse video (ATARI)** or **HELP** to get help, and **ESC** to go back to the data entry screen. Note that when you reenter the ACCESS DOS 2 Function data screen from the Help function, you must reenter any data.

Be careful that you do not become confused on this function because the **ESC** key, if pressed from the command entry level, escapes from the Access DOS 2 function and returns to the main DOS 3 menu.

The Access DOS 2 help screen is shown in the following typed example. It is typed exactly as it appears on the screen. It is the same screen you see if you select the Help function directly from the DOS 3 Main Menu. It summarizes the information given above.

**Access DOS 2 Help Screen 1 of 3**

1. Press A to access a DOS 2-formatted disk and convert one or more files onto a DOS 3-formatted disk. PRESS A WITH DOS DISK IN DRIVE 1.
2. Source drive number? Enter no. of drive with DOS 2 disk.
3. DOS 2 filespec? Wildcards are OK.

You cannot get a file index from the DOS 2 disk. But you can see the name of each file on the disk, if you

a  select Dn:*.* as filespec, and
b  use RETURN response in STEP 7.

**Access DOS 2 Help Screen 2 of 3**

4. Destination drive number? Enter no. of drive with DOS 3 disk. You can use the same or another drive.
5. DOS 3 filespec? You can use new or the same filenames. Wildcards can also be used.

**Access DOS 2 Help Screen 3 of 3**

6. Are source and destination the same disk (Y/N)? Press Y to convert files on same disk. Press N to convert files on diferent disks. (See screen.)
7. Convert all specified files (Y/N)? Press Y to convert files automatically. Press N or RETURN to convert files one at a time. DOS gives you a Y/N option on each.

Remember: Using RETURN on Step 7 with a Source Filespec of Dn:*.* causes DOS to show you the names of all files on the DOS 2 disk.

**Press P to print this screen**

**Press RETURN for more help**

If this help screen has been accessed through the main menu help function, the last line on the screen reads:

**Press ESC for help menu.**

If this help screen has been accessed while this utility was loaded, the last line reads:

**Press ESC for Access DOS 2 screen.**

# HOW TO USE THE COPY/APPEND FUNCTION

This function is accessed from the main DOS 3 menu using the C selection. It is used to make copies of files that you specify from one disk to another. It may be used with a single disk drive or with two drives. It may also be used to "append," that is, to add onto an existing file. This is, however, a more advanced function and is not likely to be used by the first-time user.

**FOR THE FIRST-TIME USER**

When you select function C, DOS 3 completes the wording for you:

**Selection? Copy/Append**

and highlights this selection on the main menu.
    It then specifies, at the bottom of the main menu:

**Loading D:COPY.UTL ...**

This means that this file name must be on the disk in drive 1 in order to be able to run this function. After the copy utility file, as it is called, is loaded, the screen is cleared and the copy utility screen is displayed. The first question asked is:

**Append? N**

In this case, the default reply already appears with the question. Press **RETURN**

again to ask the machine to accept a no-append reply. (See below for more explanation about append).

Now the next question is:

**Source device?**

Assuming that you are doing your first copy from drive number 1, press **RETURN** to make the default response appear, as follows:

**Source device? D1:**

Press **RETURN** to accept the default. The next question DOS 3 asks is:

**Source filename?**

If you do not know what a filename is, please refer to this subject in the section "What You Should Know Before Using DOS 3." In this filename, you may use the wildcards if you wish to copy more than one file, or you may specify a single file name. Note that this filename does NOT include the source device since it has already been specified. An example filename would be:

**Source Filename? COPY.UTL**

If, instead of copying just one file, you wanted to copy all files to a new disk, it performs the copy function the same as the Duplicate function if you just press **RETURN** instead of entering any file name. In this case, the default appears as: *.* which means that all files are to be copied from the source disk to the destination disk. The main differences between Copy and Duplicate are as follows:

Duplicate gives you no choice of which files to copy, but it copies all files on the source disk. "Files" are not actually copies, but rather segments of the disk in which files are stored. What this means is that disk swapping in a single-drive system is less frequent than it might be if you had used copy instead. The typical disk duplicate function, if there is 48K of RAM to use, would take 3-4 swaps of the source and destination disks in that single drive. The equivalent copy function, where all files are to be moved, requires one swap for EACH file in the file index! Therefore, if you wish to duplicate all files, the Duplicate function involves less work for you.

Duplicate also formats a disk for you if you have not already done so, all in one operation.

Copy allows you to specify which files are to be copied. If single files are specified, you may also change the name of the copy. For the copy function, however, you must have a formatted diskette in the destination drive.

The next question DOS 3 asks is:

**Destination device?**

If you have a single-drive system, press **RETURN** to make the default appear ...
**Destination device? D1:** If you have a two drive system, you may press **RETURN** to make the default appear, then use the command edit functions (CTRL + left arrow) to move the cursor left, onto the 1, and type a 2 over it.

Then press **RETURN** to ask DOS 3 to accept the input.

Now DOS 3 asks:

**Destination filename?**

The normal response is to ask DOS 3 to give the same name to each of the copies on the destination disk as were given to the files on the source. Therefore, it is normal at this point to press **RETURN** to make the default appear:

**Destination Filename? *.***

This means give all of the destination files the same name as each source file it copies.

Press **RETURN** to accept this default.

Next, if the source and destination are the same device, DOS 3 asks:

**Are source and destination the same diskette (Y/N)?**

If you are copying files from one disk to another, answer NO this question. If you are copying files onto the same disk, answer YES.

Now the system gives instructions, as:

**Insert source disk, press RETURN** (if single)

or

**Insert source disk.**
**Insert destination disk, press RETURN.**

(if a dual-disk system)

If you have a single disk system, after it reads the file (or each file if you specify more than one), it asks:

**Insert destination disk, press RETURN**

The system then informs you about the progress of the copy work, by the message:

**Copying D1:COPY.UTL to D1:COPY.UTL**

or if you have a two drive system, it says:

**Copying D1:COPY.UTL**
**to D2:COPY.UTL** (or some other name if you have changed the destination name).

**IF WILDCARDS ARE SPECIFIED**

If you use any wildcards in specifying the filename for the source disk, the copy utility asks:

**Copy all specified files (Y/N)? N**

Note that the default response N (no) is already in place.

At this point, if you press **RETURN** for each matching file name on the source diskette (a *.* wildcard matches all of them), the copy function asks:

**Copy D1:FMS.SYS to**
**D1:FMS.SYS (Y/N)? N**

In this case, the default reply is always No. For each file you wish to copy, you have to press Y, for Yes, then press **RETURN**.

If you are already sure you want to copy all files with a specific filespec or filespec with wildcards, then it is most efficient for you to respond Yes to the question:

**Copy all specified files (Y/N)? Y**

Note that you should change the N to a Y before pressing **RETURN**. Then the system copies all specified files without asking you about each one as it finds them.

After all of the matching file names have been copied from one drive to the other, it ends with the message:

**X file(s) copied**

In this case X represents the total number of names on the source diskette that matched the source filespec and were copied onto the destination diskette. In addition, if any wildcards had been used, then the message **No more** (filespec) **files found, will appear.**

This indicates that all filespecs matching the one specified have been found and copied.

The final question is:

**Do you have more files to append or copy (Y/N) ? Y**

In this case the default response is already printed with the question. If you wish to copy more, or otherwise use the copy utility program, press **RETURN**. If you change the Y to N, then press **RETURN** here, you **RETURN** to the main DOS 3 menu.

### IF YOU MAKE MISTAKES IN DATA ENTRY

If you notice a mistake in a command line before you press **RETURN**, you can edit the command using the command line edit functions mentioned in the chapter titled "What You Should Know Before Using DOS 3."

If you notice a mistake in a command line after you have already pressed **RETURN**, and wish to go back and start the entry over, the key combinations **SHIFT/CLEAR** or **CTRL/CLEAR** restart the command entry at the first line again.

### HOW TO GET HELP

This utility has a built-in Help function, accessed by pressing the inverse video (ATARI) key or the **HELP** key. You can access this help feature to read some useful information about this function if you cannot remember what to enter at a particular point. To return from Help, to continue your data entry on the copy screen, press the **ESC** key.

You may go back and forth between the data entry screen and the Help screen as often as you wish, pressing the inverse video (ATARI) or **HELP** key to get help and the **ESC** key to return to data entry. However, as with the duplicate function, be careful not to become confused. An **ESC** key, if hit from the copy entry screen, escapes from that function and returns you to the main DOS menu. Note, however, that when you reenter the Copy function data screen from the Help function, you must reenter any data. Data is not saved when you ask for help. A copy of the help screens for this function is contained at the end of this section. They summarize the information contained in the section.

### FOR THE ADVANCED USER

The Copy function allows you to copy files from any legal source device to any legal destination device. If the device is not a disk, then the source or destination filespec question will not appear. This means that you may, if you desire, use this utility to copy files to the screen, to the printer, or any other device for which a handler is present.

### APPENDING FILES

The Append function allows you add onto the end of an existing file. All of the questions are the same as those already asked during the function, however, instead of a separate file being generated for the copy, the new data is simply "tacked onto the end" of an existing file. Note that this file must exist or the append function generates an error (file not found).

### THE COPY FUNCTION HELP SCREENS

These screens are the same ones you see whether you access them from the HELP function of the main DOS 3 menu or ask for Help during the command entry mode. They are as follows:

**Copy/Append Help Screen 1 of 4**

**1. Press C to COPY or APPEND files. COPYing creates a duplicate file on the same or another disk. APPENDing copies a file and joins it to another file. PRESS C WITH DOS DISK IN DRIVE 1.**

**Note:** Use COPY to write DOS files onto other diskettes as needed.

2. **Append (Y/N)?** Press Y to append (join) two or more files. If you press N or RETURN, DOS will copy without appending.
You can COPY or APPEND using one or two disks and one or two drives.

**Copy/Append Help Screen 2 of 4**

3. **Source device?** Specify device with source file(s). Dn: = drive no., E: = screen, C: = cassette.

4. **Source filename?** Enter filename. Wildcards can be used.

5. **Destination device?** Specify device to receive file(s) or with file(s) to be appended. Dn: = drive no. E: = screen, P: = printer.

**Copy/Append Help Screen 3 of 4**

6. **Destination filename?** Wildcards can be used. If appending, use existing name. If copying onto same disk, choose new filename.

**Note:** DOS does not allow duplicate filenames on a single diskette.

**Copy/Append Help Screen 4 of 4**

7. **Are source and destination the same diskette (Y/N)?** Press Y to copy/append files on the same disk. Press N to copy/append files on different disks.

8. **Copy/Append all specified files (Y/N)?** Press Y to copy/append all files in source filename. Press N or RETURN to copy/append files one at a time.

9. **When processing files one by one, DOS gives you a Y/N option for each.**

Press P to print this screen
Press RETURN for more help

If this help screen has been accessed through the main menu help function, then the last line on the screen reads:

**Press ESC for help menu**

If this help screen has been accessed while this utility has been loaded, then the last line reads:

**Press ESC for Copy screen**

# HOW TO USE THE DUPLICATE FUNCTION

This function is used to duplicate a complete disk. The disk from which the copy is to be made must be one which has been formatted using DOS 3. It is called from the main DOS 3 menu using the **D** selection. On selecting D, the system command line specifies:

**Selection? Duplicate**

Then this same word is highlighted in the menu selection area. It places the following line at the bottom of the menu screen:

**Loading D:DUPDISK.UTL ...**

This means that your diskette must have this file name on it in drive 1 in order to execute the duplicate disk function. Now the screen clears, and presents the Duplicate information screen. Next the system asks the question:

**Source drive number?**

For the first-time user making a DOS 3 backup (or a user making a backup of any kind of disk)

1. If you have a single drive system, press **RETURN** 3 times. This results in the following display:
**Source drive number? 1**
**Destination drive number? 1**

2. If you have a two drive system, and you wish to duplicate from drive 1 to drive 2, then follow instruction number 1 above. Then press the Backspace key, and type over the 1 at the Destination Unit question, to make it a 2.

3. Now press **RETURN** again.

4. The computer now gives one of two possible responses, depending on whether a single or a dual drive system was indicated. For example, it may say either:
**Insert source disk in drive 1**
**Insert destination disk in drive 2**
**Press RETURN**
(if you have a two-drive system)
or
**Insert source disk in drive 1**
**Press RETURN**

5. Be sure your source diskette has been write-protected. This means that the notch along the side of the diskette has been covered by a write-protect tab. Your ATARI DOS 3 Master Diskette is formed without a notch, so it is already write-protected.

6. From here onward, you may follow the instructions that appear on the screen about handling of the source and destination disks.

Note that if you are using a two drive system, it is not necessary to handle the disks until the duplication is completed. Use of a single drive system requires that the source disk be inserted to read the source materials into memory, then the destination disk has to be inserted to write these materials onto it.

As the copy progresses, the computer tells you about the progress, showing how many blocks remain to be read and how many remain to be written.

**HOW TO GET HELP**

This function is one with a built-in Help feature. You can, at any point in the command entry sequence, see the Help screen by pressing either the Help key on your ATARI series XL computer, or the **ATARI** key [ ⅄ ] on your ATARI 400 or 800 computer.

Here is an illustration showing the appearance of the duplicate disk help screen, as it appears when you push the HELP or inverse video key:

**Duplicate Help Screen 1 of 1**
**1. Press D to copy all files in same location onto a new disk—to make an exact"logical"duplicate disk. PRESS D WITH DOS DISK IN DRIVE 1.**
**2. Source drive number? Enter no. of drive with original. CAUTION: PUT WRITE-PROTECT TAB ON ORIGINAL.**
**3. Destination drive number? Enter no. of drive with new disk. You can use the same or another drive.**
**4. Are source and destination the same diskette (Y/N)? Press Y to duplicate files on the same disk. Press N to duplicate files on different disks.**

**Press P to print this screen**
**Press RETURN for more help**

If this help screen has been accessed through the main menu help function, the last line on the screen is:
**Press ESC for help menu.**

If this help screen has been accessed while this utility was loaded, the last line reads:
**Press ESC for duplicate screen.**

## THE MEANING OF THIS HELP SCREEN

Notice that this is the same help screen you see when you use the Help function from the main menu when you ask about the Duplicate Function.

The help screen tells you that a DOS disk must be in drive number 1 before this function works (as mentioned, it needs the file named DUPDISK.UTL).

Item 2 shows you the correct reply to the source drive number. Notice that it does say number rather than "device". This means that you must say

**1** or **2** or **3** etc,

specifying the drive numbers to use. You do not specify"D1"or"D1:"here.

Notice at the bottom of this Help menu, it says

### "Press ESC for Duplicate Screen"

What this means is that pressing **ESC** (once) redisplays your work screen for the Duplicate function. Although you may switch back as often as you want between Help and the data entry screen, note that **ESC** takes you back to the initial prompt and all data input on the screen is lost.

Be careful not to become confused when you press the **inverse video** or **HELP** key (to go to help), and **ESC** (to escape from help). If you press **ESC** while you are in the Duplicate function screen, you escape from duplicate! This returns you to the main DOS 3 menu.

On the help screen, item 2 also cautions that you should use a write-protect tab on the source diskette. This is for your own protection. In the possible event of a command entry error, your source diskette cannot be accidentally written over.

This precaution is not required for your DOS 3 Master Diskette in that it has no write-protect notch, and is always write protected. Note however that you should normally be using a copy of your master diskette, rather than the original (except of course for the first copy ever) to protect it. In any case, you may wish to write protect whatever source copy you are using to prevent accidental loss of data.

Section 3 of the Help menu tells you what to do when you get the second question, the destination drive number. You may make duplicate copies of a disk using either one or two drives. If you have only one drive, then the responses are:

**Source drive number: 1**
**Destination drive number: 1**

These are the default responses, so you can, using the main Duplicate-disk screen, press **RETURN** three times to get the screen to read as it shows above.

This working copy can then be used for all of the examples used throughout this manual. Note that a copy of the master is REQUIRED if it is to be written onto. This is because, as specified above, the Master Diskette is permanently write protected.

The Duplicate screen tells you what to do. Refer to the Duplicate-function menu. (Remember it is **inverse video/HELP** to go to help, and **ESC** to escape from HELP). If you have only 1 disk unit, you have to specify the following:

**Source drive number? 1**
**Destination drive number? 1**

Note that as each of the command question lines appears, you may either press **RETURN** to make the default response appear, then RETURN again to accept the default, or you may enter your own response. You also have the option, after the default appears, of using the Command Edit function (see "What You Should Know Before Using DOS 3") to change the data to match what you really want to do before pressing **RETURN**.

Now the system leaves your responses printed on the screen, and tells you:

**Insert source disk in drive 1**
**Press RETURN**

Once you press RETURN, it reads the source disk, then replies:

**Insert destination disk in drive 1**
**Press RETURN**

If the destination disk is found to be unformatted, the initialize disk function is automatically performed before the destination data is written. During the term of this single-drive copy, system memory is used as the buffer (temporary storage place) for the data. DOS provides status information on the duplication with a series of messages such as:

**59 blocks remain to be read**
**59 blocks remain to be written**

This indicates where the system is in the process of the duplication. The blocks to be read are coming from the source diskette. The blocks to be written are going to the destination diskette.

If you are using a single-disk system for the duplicate, you may be required to swap the diskettes many times. Therefore, the write-protect tab on the source diskette is especially recommended if you happen to"slip"once. By accident you may have the source disk in the drive when it is supposed to be the destination. This could destroy your source data if it is not write protected. Write protect is less critical when a two-drive system is used because no diskette swapping is required. It is still a good idea, however, that the source be write protected. In the two-drive duplicate process, all duplication is done automatically and proceeds to the end with no swapping required.

### ADVICE FOR THE NEW USER

Once the new user has a working copy (also called a System Diskette) of the ATARI DOS 3 Master Diskette, the master should be put away in a safe place, and only the backup copy should be used. If desired, multiple copies may be made in this manner, but of course only enough for one's own personal use.

# HOW TO USE THE ERASE FUNCTION

The purpose of the Erase function is to remove one or more files from the disk. By removing files from the disk, you are telling DOS 3 that the space occupied by those files may be used for other purposes. It is a way of making room for other files by deleting those not needed any longer.

NOTE THAT ONCE YOU TELL DOS 3 TO ERASE A FILE, THERE IS NO WAY TO REVERSE THE PROCESS.

The Erase function is entered by using the first letter E from the main DOS 3 menu. On entering this function, DOS 3 completes the word **Erase** opposite the word **Selection** on the command line. It also highlights this selection in the menu area.

Next it asks the question:

**Filespec?**

To ask which file or files you wish to erase. Wildcards may be used in the filespec.

### FOR THE FIRST-TIME USER

The following set of instructions provides a new file on the disk that you may then erase with this function. Note that this instruction sequence can only be done with a disk which is NOT write protected. This means you cannot perform this function on your ATARI DOS 3 Master Diskette, but you can use a copy of the master which you have made as a backup disk. Simply be certain that the disk you use is not write protected.

1. From the Main DOS 3 menu, select function F, File Index function.

2. When the computer asks
   **Filespec?**

   Press RETURN twice. This makes the default appear as
   **Filespec? D1:\*.\***

   and tells the computer to accept the default.
3. When the computer asks
   **Display Device?**

   Type D1:FILES
   and press RETURN
4. The file index appears on the screen and is stored in the disk file named FILES on disk unit 1.
5. Confirm that this name is present on the disk by performing the file index function again. This time, select function F, as before, but press RETURN 4 times to accept all of the defaults. Now the file index appears on the screen and includes the new name called FILES.
6. Now you can select the Erase function.

## WHEN DOS 3 ASKS ABOUT FILESPEC?
If you press RETURN, it shows the default filespec of D1:\*.\*. If you again press RETURN to accept the default, it tells DOS 3 you want to either Erase all files on the disk, or DOS 3 presents all of the file names, one at a time, asking individually for each file if the file is to be erased.

   If you already know the name of the file you wish to erase, type it as the filespec, for example:
**D1:FILES**
or
**D:FILES**
or
**FILES**

All of which are equivalent. This is because D alone assumes the default D1. Likewise, the name alone assumes the default of D1. You may also specify wildcards, in place of the D1:\*.\* you could specify
**FI\***

Here DOS 3 presents to you for decision all files on drive 1 that begin with FI and end in any manner.

## WHEN DOS 3 ASKS ABOUT ERASING ALL SPECIFIED FILES
The next question asks if you wish to
**Erase All Specified Files (Y/N)? N**

Note that the default reply N is already in place
   If you press RETURN at this point, it means that you wish DOS 3 to ask you whether this is the file you want to erase each time a file name matches the filespec you have given. Then for each file match, you see the message:
**Erase D1:FILES. (Y/N)? N**

This means that to actually erase the filename, you need to take positive action, changing the N to a Y, and press RETURN. This protects you from a possible error by giving you a last chance to change your mind.
   When DOS 3 asks about erasing all files with matching filespecs, if you change the N to a Y, for Yes, it means just go ahead and erase all files matching the filespec. In other words, don't ask questions, just do it.

### HOW DOS 3 REPORTS PROGRESS

As it accepts your instructions that a file is to be erased, it reports on the screen:
**Erasing D1:FILES**

Next it reports on the status line that the specified file name has been erased. Note that DOS 3 does not erase protected files. Then, once all matching files have been found, questioned, and bypassed or erased, it reports completion status on the status line of the main menu as:

**Job Completed. n Files erased**

The **n** represents the total number of files actually erased.

### IF YOU NEED HELP

While the computer is asking you for information, you may press the **ATARI** key or the HELP key to see the following summary of the Erase entry instructions:
**Erase Help Screen 1 of 1**
1. **Press E to delete one or more files from your disk. (DOS will not erase Protected files.)**
2. **Filespec? Specify files to delete. Wildcards can be used. CAUTION: ERASED FILES CANNOT BE RECOVERED.**
3. **Erase all specified files (Y/N)? Press Y to delete all files named by filespec. Press N or RETURN to erase files one at a time.**
4. **If erasing one by one, DOS gives you a Y/N option for each file. Erased filenames should no longer appear in file index.**
   **Press P to print this screen.**
   **Press RETURN for more help.**
   **Press ESC for HELP Menu.**

After you have read the Help information, press **ESC** twice to return to the main DOS 3 menu. Then you may restart the ERASE function.

# HOW TO USE THE FILE INDEX FUNCTION

This command is used to view the list of the files stored on a diskette. It is selected from the main DOS 3 menu using the first letter F.

It allows you to specify the file names you are looking for and list only those you wish. As an alternative to simply listing the file index on the screen, you may direct it to be printed out, or placed on the disk as a separate file. These options are:

### FOR THE FIRST-TIME USER

The most common thing you do is to list the contents of the diskette in disk drive number 1. This is what all of the system defaults are set to do, so this is the first example:

LIST TO THE SCREEN THE CONTENTS OF DISKETTE IN UNIT 1
When you type **F**, DOS 3 responds by typing the rest of the command

**File Index**

opposite the word
**Selection?**

At the same time, it highlights the selection in reverse video in the menu selection portion of the display.

On the next prompt line, it asks:
**Filespec?**

At this point, the simplest set of replies accept all of the system defaults. That is, to list all of the files residing on disk drive 1. This consists of pressing **RETURN** four times. The system does the rest.

Examine what the system places on the screen each time you press **RETURN**:

**Selection? File Index**
**Filespec?** (before you press **RETURN**)

**Filespec? D1:*.***
(after pressing **RETURN** once)

**Filespec? D1:*.***
**Display Device?**
(after pressing **RETURN** a second time, it has used the default filespec showing D for disk unit, 1 for unit #1, and *.* for a wildcard, saying to list all files which match this combintion. Since a wildcard is used before and after the (.) it matches ALL filenames. Therefore, all file names appear in the file index list.)

**Selection? File Index**
**Filespec? D1:*.***
**Display Device? E:**
(after pressing RETURN a third time. This says that the default display device is **E:** This means to send the display to the screen.

Now if **RETURN** is pressed a fourth time, accepting all of the default conditions, the screen is cleared, and the file index is sent to the screen. If there are more than 18 files to be listed, then at the bottom of the screen is the following instruction, called a "prompt":

**Press RETURN for more file index**

When you have displayed the final page of the file index at the bottom of the page, it states:

**Press RETURN for menu**

This returns you to the main DOS 3 menu. The file index display itself shows you a number of things about the diskette in the selected disk unit. First, it shows the filenames of all of the files and the "extensions" (see the section titled "What you should know before using DOS 3" if you are unfamiliar with this term). It also lists the amount of space in "blocks" which is taken up by each one of these files. A block is 1024 bytes (memory locations).

If there are any "protected" files on the disk, there is an asterisk (*) in the far left-hand column of the file index for that filename. For more information about protected files, see the section titled "How to Use the Protect Function."

Finally, the file index tells you how many free blocks there are on the disk. This information lets you determine if there is going to be enough room left on the disk to store the next set of programs you are working with. If you already know that a program takes up about a certain number of blocks, you can figure out if it fits. If there are no blocks free, you cannot store any more files on a disk. To make areas free again, you have to use the Erase function.

For the first time user, it would be best, at this time, to make a backup copy of your Master Diskette. Turn now to the section titled "How to use the Duplicate Function" to find out how to do this. Then, if you wish, you may return to this section or simply proceed to other segments of the manual as you need to do certain functions.

**OTHER OPERATIONS POSSIBLE WITH FILE INDEX**

When the menu specifies:

**Selection? File Index**
**Filespec?**

In this case you may specify your own filespec instead of pressing **RETURN**. For example, you could specify disk unit number 2, as:

**Filespec? D2:**

To list all of the files on drive 2. Note that for the file index function, when the filespec is accepted by the **RETURN** key, it fills in the wildcards *.* if it finds only the filespec   D1: which makes it exactly equal to   D1:*.*

### USE OF WILDCARDS

If you specify the wildcard *.*, DOS 3 is going to list the names of all of the files, regardless of their names. If you are looking to see if a specific file is on the diskette (at your request) it lists only those that have certain features. For example:

**Filespec? *.BAS**

Note that this command lists only files that have a .BAS as their file type extension. Notice in this example that there is no mention of the disk unit number in the filespec. DOS 3 assumes that the device to be used is a disk, and that the unit number is 1; it assumes this unless told otherwise.

The other is the single character wildcard (see the section on wildcards in "What you should know before using DOS 3"). For example:

**Filespec? H?LL.BAS**

This would match the names HALL.BAS and HILL.BAS for example. Therefore such names, if present on the disk, would appear in the file index list if the command was given.

### REDIRECTING THE FILE INDEX LIST

DOS 3 allows the file index listing to be directed to any device. The most common destinations for the file index are the screen (E:) and the Printer (P:). Since the screen is by far the most common, it is the default. Specifying P: before you press **RETURN** directs it to the printer. You may, if you wish, direct it to any other device, including a disk file if you desire. In this case, you might specify, for example:

**Filespec? D1:*.***
**Display Device? D2:DISK100.NDX**

This would form a new file named DISK100.NDX on the diskette in drive 2, containing the index of the disk in unit 1. This is a way of forming a kind of a catalog that you might later use to keep track of what program is on which disk.

Note that in each case where you have pressed **RETURN** to answer a question, such as:

**Filespec?**

or

**Display Device?**

and have received on the screen the default, you do not need to accept the default condition. You may

1. use the editing keys to move the cursor, and change the default to what you need, or
2. use the **SHIFT** + **CLEAR** to restart the command, or
3. use the **SHIFT** + **DELETE** to erase the default and restart this part of the command only, or
4. use the **ESC** key to return to the main menu.

## IF YOU NEED HELP

While the computer is asking you for information, you may press the **ATARI** key or the **HELP** key to see the following summary of the File Index entry instructions:

**File Index Help Screen 1 of 1**

1. Press **F** to select this function. The index tells you which files are on your disk.
2. Filespec? Specify files you are looking for. To see all files on disk, press **RETURN**. D1:*.* is the default response.
3. Display device? Press **E:** or **RETURN** to list Index on screen. Press **P:** to print Index. Index lists filenames, extenders, file sizes in blocks and number of free blocks left.
4. If list exceeds one screen, press **RETURN** for more files.

   Press **P** to print this screen
   Press **RETURN** for more help
   Press **ESC** for **HELP** menu

After you have read the Help information, press **ESC** twice to return to the main DOS 3 menu. Then you may restart the File Index function.

# HOW TO USE THE GO-AT-HEX-ADDR FUNCTION

This function is used normally by the assembly language programmer. It is selected by the letter **G** from the main DOS 3 menu. On selection, DOS 3 completes the word Go-at-hex-addr opposite the word Selection, and highlights this selection in the menu area.

   This function is called to direct DOS 3 to transfer control to a specified hex address. It is normally used to restart a program which was placed on the disk using the SAVE function of DOS 3.

   This function is used either if you SAVEd a program specifying a RUN address, or for which you tell DOS 3 during the Load, that it is NOT to RUN this program. When it specifies:

**Run addr (Hex)?**

This specifies a hex address in the range 0000 to FFFF. On pressing **RETURN**, it immediately transfers control to the specified address. Note that this address must contain executable code, or else unpredictable results occur.

**Example:** If you wish to try this function, specify E477 and press **RETURN**. This reboots the system.

## IF YOU NEED HELP

While the computer is asking you for information, you may press the inverse video key or the **HELP** key to see the following summary of the Go-At-Hex-Address entry instructions:

**GO-AT-HEX-ADDR HELP SCREEN 1 OF 1**

1. Press **G** to enter RUN addr for assembly language program in RAM.

   You will need to use GO if you SAVE
   your program file with no RUN addr.

2. Run addr (HEX)? Enter hex RUN addr. CAUTION:
   ADDR MUST CONTAIN EXECUTABLE CODE. OTHERWISE, ERRORS REQUIRING REBOOT MAY RESULT.

**Press P to print this screen**
**Press RETURN for more help**

After you have read the Help information, you have to press **ESC** to return to the main DOS 3 menu. Then you may restart the Go-at-hex-addr function.

# HOW TO USE THE HELP FUNCTION

This function is selected from the main DOS 3 menu by the letter **H**. Its purpose is to provide you with an "online" or immediately available way of accessing helpful information about using DOS 3. On selecting this function, DOS 3 highlights this function name in the menu area, then completes the word Help, as:

**Selection? Help**

On the bottom section of the screen, it states:

**LOADING D:HELP.UTL ...**

This means that this file must be present on the disk in drive 1 in order to run this function.

Next, the screen is cleared, and a duplicate of the main menu, this time titled:

**HELP MENU**

Beneath that new menu, it states:

**Help On Which DOS function? H**
**Press H for general information**

In the question line, the default, **H**, is already present. Therefore, if you simply press **RETURN**, the system provides you with six different screens specifying general information about the use of DOS 3. These screens are printed within this section. Further information about the data covered in these screens may be found in the section titled "What You Should Know Before Using DOS 3."

You may specify any of the other functions, to read and/or print their help screen information. Each time you make a selection, the Help menu indicates:

**Help on the way ...**

This display shows while it is searching for the correct information screen. This information is taken from a file on the Master Diskette called HELP.TXT. Therefore, to get help from the Help function, this file must also be on the disk contained in drive 1.

When it presents the Help screen to you, you have a choice for each:

Print the screen (your printer must be on), **RETURN** to see the next screen if there is more than one, otherwise the current one is redrawn on the screen, or **ESC**ape to the Help menu again for another possible selection.

From the main Help menu, you may press **ESC** to return to the main DOS 3 function menu or select another item for which to view its help information.

The Help screens for each of the DOS 3 functions have been placed, throughout this manual, in the sections where the function itself has been described. The screens contained immediately following are the general information Help screens mentioned above.

(The rest of the Help screens are listed under a subtitle "If you need help" or "How to get Help" in the sections describing the individual DOS 3 functions.)

## GENERAL INFORMATION - 1 OF 6

The DOS commands allow you to store, retrieve, and manage your disk files.

1. To select a menu function, press the first letter of the command name. (Erase ... press E, etc.)

DOS prompts you to supply the data needed to execute a command. When response is complete, press RETURN.

2. To get Help on any DOS function, press the inverse video or HELP key instead of typing in requested data.

3. To break out of any DOS function, press ESC key for the DOS menu.

Press P to print this screen
Press RETURN for more help

## GENERAL INFORMATION - 2 OF 6

Some DOS commands are stored on the DOS disk and are loaded into the computer when selected.

    Thus, keep DOS disk in drive 1 when selecting:

1. Copy/Append
2. Duplicate
3. Init disk
4. Access DOS 2
5. X-user-defined
6. Help

These files are stored on the DOS disk to leave you more available memory in your computer.

Press P to print this screen
Press RETURN for more help
Press ESC for Help menu

## GENERAL INFORMATION - 3 OF 6

DOS asks you for data such as:
1. Filename? The full name of your file. (MYFILE.BAS)
2. Filespec? The filename plus the drive ID. (D1:MYFILE.BAS)

Wildcards can be used for 1 and 2.
= Any combination of characters.
= Any single character.

3. Device: Dn:= drive no.; E:=screen; P:=printer; C:=cassette.
4. Source—? From which—?
    Destination—? To which—?

Press P to print this screen
Press RETURN for more help
Press ESC for Help menu

## GENERAL INFORMATION - 4 OF 6

Answers to prompts sometimes appear highlighted (Drive number ?1). DOS picks these preset values (defaults) for you when you press RETURN.

The default filespec is D1:*.* (all files on the diskette in drive 1).

When using wildcards, DOS asks you if you want to act on all files:   function all specified files (Y/N)?

1. Press Y to process all files automatically. Press N or RETURN to process files one at a time.

**GENERAL INFORMATION - 5 OF 6**
As noted in the previous screens, the RETURN key is used:
(a) To signal the end of your response to a prompt, and
(b) To select the default value in response to a prompt.

There are two more points to note:
1. When in doubt, RETURN is safe. If you answer RETURN to a prompt, DOS never does anything to your files.
2. When DOS asks (function) all specified files (Y/N)? Press RETURN to see each filename in filespec.

**GENERAL INFORMATION - 6 OF 6**
You know about using the HELP or inverse video, ESC, and RETURN keys. But there are four other special uses for the following keys:
1. SHIFT-CLEAR causes a DOS function to restart from the first prompt for data.
2. SHIFT-DELETE deletes your response to the current prompt only.
3. BACK SPACE erases the character preceeding the cursor.
4. CTRL /CTRL moves the cursor without erasing any characters.

# HOW TO USE THE INIT FUNCTION

This function is selected by pressing I from the main DOS 3 menu. When I is pressed, the system command line specifies:
**Selection? Init disk**

Then highlights these words in the menu selection area. It places the following line at the bottom of the menu screen:
**Loading D:INIT.UTL ...**

This means that this file name must be on the disk in drive 1 at the time you try to execute this function.
    The Init function of DOS 3 is used to prepare a disk for use. It writes "format" information on the disk as described in the section titled "Format" in the chapter titled "What You Should Know Before Using DOS 3".

**FOR THE FIRST-TIME USER**
The function most likely to be used by the first-time user is to format a disk in disk unit 1. This example shows this action.

1. To format a disk, you must first be sure that the write protect notch is NOT covered. This allows the computer to write new information onto the disk.
2. When the Init Utility program has been loaded, the screen appears as follows:

The first line says
**Format diskette in drive (1-8)?**

Press RETURN to make the default response appear, as:
**Format diskette in drive (1-8)? 1**

This is the drive where you place your new disk, which is to be formatted. If you have a single-disk system, it is always drive 1. When the question first appears, the cursor is at the end of the line, waiting for you to specify the drive number. If you press **RETURN** once, the number 1 appears. You may change it, and initialize a disk in a different drive if you wish. Then press **RETURN** again to ask the computer to accept the number you have entered as the drive to initialize.

Now the computer asks:

**1 for single density**
**2 for double density**

If you press **RETURN**, the default 1 appears. This is because this format is the only one the ATARI 810 Disk Drive may perform, and the only one that may be read by all ATARI disk drives. If you intend to exchange software with any other ATARI computer owner who uses an 810 disk drive, this is the format you should choose. Next the computer asks:

**Write FMS.SYS files? N**

This line shows the default response, N, that you get on the screen if you just press **RETURN**. The other possible response is Y. This says that the diskette you are initializing should be a System Diskette, capable of "booting" DOS 3. Use the command edit functions to modify the line to read Y or N, then press **RETURN** to ask the computer to accept that response (Y for system disk, or N for data disk).

If you have selected Y as the response to the above, then the following line appears:

**Modify FMS parameters? N**

This line is shown with the default response you get by pressing **RETURN** once. This is the normal response. Press **RETURN** to accept it. The advanced user should refer to the section titled "Modifying the File Manager" below for more information about the other option (Y) here. Now the computer clears the screen, and summarizes for you just what you told it to do.

**CHANGING PART OF A COMMAND**

If you have entered a command incorrectly, you may restart that command by pressing **SHIFT/CLEAR** at any time during your command entry in the Init function.

For information about other ways of changing the commands you have entered, see the section in this manual titled "Command Editing."

**RETURNING TO THE DOS 3 MENU**

If, during the command entry mode, you decide you don't want to perform this function after all, you may press the **ESC** key. This reloads the DOS 3 main menu.

**TO START THE FORMAT**

Follow the instructions given at the bottom of the summary page. When you press **RETURN**, the computer informs you about the progress of the format operation, with the line:

**Now formatting diskette ...**

and finally the lines

**Initialization completed.**
**Press RETURN to init next diskette**
**Press ESC to return to DOS**

At this point, if you just press **RETURN**, the summary page is restored to its original appearance. This is a continue operation to allow you to initialize more diskettes in exactly the same way in the same disk unit if you wish, without going through the data entry process again. If you press **ESC** you return to the main DOS 3 menu.

## MODIFYING THE FILE MANAGEMENT SYSTEM

The command response Y to the question

**Modify FMS parameters?**

brings up the following responses. The default replies are shown in the sample lines below, with an explanation of what each one means. Modifications to these items is usually done by a more advanced user.

**Starting address of FMS buffers? 1A7C**
**Number of system buffers (2-16)? 5**
**Verify write commands (Y/N)? Y**

Taken in the order they are shown:

**Starting address of buffers? 1A7C**

The file manager (FMS) maintains buffers for the data which is to be sent to or received from the disk or any other unit which it is controlling. This address specifies where the buffers begin. A user may choose to relocate them up in a higher area of memory in order, for some reason to use this particular space allocated for the standard FMS, however the size of the buffers (see below) determines exactly where the starting address may be defined.

**Number of system buffers? 5**

This specifies the total number of disk units plus the number of files open at any one time, whether for input or output. Each system buffer is 128 bytes long. This limits the number of OPEN commands that may be issued to the FMS.

**Verify write-commands? Y**

This is the normal response. Yes says that when a disk write is issued, the system should read the data back again, to verify that it was written OK and can be read again later. If it is not OK the first time, this allows the system to put the same data somewhere else on the disk and mark the unreadable area as bad. A user usually selects the N option to speed up the data transfer to the disk, but in the process is taking the chance that a disk write error may cause the data to be unreadable later. It is up to the user to balance the needs for speed against the possible chance of problems in disk reading.

When all these options are selected, they appear in the summary display. As with all other options, pressing **RETURN** once without typing anything makes the default option appear. Pressing **RETURN** again with the default or the actual selection present tells Init to accept the data, and go on to the next function.

## HELP FOR THIS FUNCTION

A Help screen for this function may be viewed by pressing the **inverse video** key or the **Help** key. The Help screen for this function is the same one you see if you select the Help function from the main menu. However, this Help screen may be accessed back and forth from the Init function without exiting Init. You always press the **inverse video** key or **HELP** key to enter Help for this function, then the **ESC** key ONCE to exit to the data entry menu. You may go back and forth as many times as you wish. But be careful not to press **ESC** from the data entry menu, since this exits the function and returns you to the main DOS 3 menu.

The Init Help screen contains a summary of the information specified in this section, and appears as follows:

**Init Disk Help Screen 1 of 2**

**1. With DOS Disk in Drive 1, press I to initialize disk.**

**CAUTION: Initializing a disk erases all data on the disk.**

2. Format diskette in drive (1-8)? Enter no. of drive with disk to be initialized. **NEVER INIT YOUR DOS DISK.**

3. Format type? Press 1 for single-density format; press 2 for double-density format.

Init Disk Help Screen 2 of 2

4. Write FMS.SYS file (Y/N)? This file lets you boot the system from this disk.

   Press Y to write file; press N or RETURN to format only.

5. Modify FMS parameters (Y/N)? This lets you increase usable memory. Press Y to modify; press N or RETURN to bypass modification.

If you press Y for Step 5, answer:
a) Start address of FMS buffers?
b) Number of FMS buffer (2-16)?
c) Verify write – commands (Y/N)?

DOS asks you to confirm init values.

Press P to print this screen
Press RETURN for more help

If this Help screen has been accessed through the main menu help function, the last line on the screen reads:

**Press ESC for Help menu**

If this Help screen has been accessed while this utility was loaded, the last line on the screen reads:

**Press ESC for init disk screen**

# HOW TO USE THE LOAD FUNCTION

This function is called from the main DOS 3 menu by typing L. It is used to bring back, into memory, data which was sent to the disk using the Save function. It is intended for the advanced user.

When you type L, it makes the selection line appear as follows:

**Selection? Load**

and, at the same time, highlights the Load function in the main menu.

It then asks:

**Filespec?**

requesting that you specify which device and filename holds the file you wish to load. A valid example is the one which was used in the example of the Save function.

**Filespec? D1:MEM3000    RETURN**

The system then searches the source device and gives you another chance to think about whether this is the file you wanted. (Wildcards can be used in this filespec, so you may or may not want to load the file the system presents to you as a possible choice.) When DOS 3 has found a file that matches the filespec you have given, the system will present a question at the bottom of the menu. This question it is asking tells you that it has indeed found a file which matches the filespec you have given.

In the example, the system would present the question at the bottom of the menu:

**Load D1:MEM3000 (Y/N)? N**

with the default N already on the screen. If you just press **RETURN** here, the file is not loaded. If you change the N to a Y, for Yes, the file is loaded.

If there is an INIT address associated with the file it loads, DOS 3 performs a JSR to it for initialization. If there is a RUN address which had been saved with the file, DOS 3 could JSR to it following the optional INIT routine. See the explanation of INIT and RUN addresses in the section of the Save function. If there is a RUN address Saved with this file (see the DOS 3 Save function), then DOS 3 asks, after performing the INIT function, whether this newly loaded program or memory segment is to be run by the question:

**Run this file (Y/N)? N**

Note that the default, No, is already printed on the screen. If you do wish to RUN it, change the N to a Y, for Yes, then press **RETURN**.

If a wildcard has been used in the filespec, and if there is more than one file which matches this filespec on the source device, you may continue to answer Yes as the system asks:

**Load** (name of file you typed in) **(Y/N)? N**

for the loading of each one. Loading terminates, however, with the load of the first file which has a RUN address Saved within it, and for which you specify that you do indeed want to run this program. This is because DOS 3 loses control over the system at this point in favor of granting control to the loaded routine.

If you have chosen not to RUN the program which was loaded and elect to start the program later, after more DOS 3 operations, you may use the G function, Go-at-hex-address to start the program. See this function for more details.

**IF YOU NEED HELP**

While the computer is asking you for information, you may press the inverse video key or the **HELP** key to see the following summary of the Load entry instructions:

**Load Help Screen 1 of 1**

1. **Press L to load into selected memory area a previously saved assembly language program.**
2. **Filespec? Enter filespec of file to be loaded. Wildcards are OK. If wildcards are used, DOS shows you each filename and asks you which file to load.**
3. **After selecting file to be loaded, DOS asks Run this file (Y/N)? Press Y to run the file. Press N or RETURN if not to run the file.**

**Press P to print this screen**
**Press RETURN for more help**

After you have read the Help information, press **ESC** to return to the main DOS 3 menu. Then you may restart the LOAD function.

# HOW TO USE THE MEM.SAV FUNCTION

This function allows you to change DOS 3 so that it automatically saves certain memory areas in a file in drive 1 called MEM.SAV. This function allows you to use many of the DOS 3 functions without disturbing a program which you may be currently working on in the command entry mode in ATARI BASIC.

DOS 3, when called, saves the content of lower memory into a file named MEM.SAV. This is the memory space that DOS itself needs to perform most of its normal functions.

Basically, when you boot DOS 3 with a cartridge present, the files KCP.SYS and FMS.SYS are loaded. These two files are the ones which perform most of the normal functions of communicating between you and the disk. They handle all of the normal commands that you can issue from BASIC and operate in memory along with BASIC. When you want to do a DOS 3 menu command, you must type DOS and press **RETURN**. DOS 3 at this point loads the file named KCPOVER.SYS in order to write the menu and to be able to understand the menu commands.

If the function MEM.SAV is active, it either creates or replaces a file named MEM.SAV on disk 1. This is used to store temporarily the current contents of the memory space it needs to use for KCPOVER.SYS. Then, if no more space is needed during the performance of any of the DOS commands which you use, DOS 3 can use this MEM.SAV file to restore the original contents to this memory area when you perform the To-Cartridge function.

This function either enables or disables the MEM.SAV option. When you enable MEM.SAV, it creates or uses a file on disk unit 1. Therefore this disk cannot be write protected if this function is used. When you enable MEM.SAV, it deletes the file.

If MEM.SAV is disabled when you type DOS, the DOS control files are loaded into memory and erase your program. When you return to the cartridge, the existing program is gone.

Note that it takes a few seconds either to go from BASIC to DOS or from DOS to BASIC if MEM.SAV is used. This is because it takes some time to write or read the 5 blocks that MEM.SAV uses. You must decide whether it is necessary to have this autosave of the lower memory. If you don't often use DOS right in the middle of program entry, you probably want to keep MEM.SAV off. This saves the time it takes to read or write these files as you go from BASIC to DOS and back.

## HOW THE FUNCTION IS CALLED

Press the M key from the main DOS 3 menu. DOS 3 completes the function name MEM.SAV opposite the prompt name Selection. It also highlights this selection in the menu area. Then it asks:

## Use MEM.SAV (Y/N)?

Note that the default response, the current setting, is already present. If you press **RETURN** at this point, MEM.SAV status remains the same. If you change the response and then press **RETURN**, the MEM.SAV status is changed accordingly.

You can tell if it is currently active or not by looking in the DOS 3 Menu area. The word includes a dot, for example:

## MEM.SAV

If the dot is shown, the function is active. In this case, the lower memory is restored when going back to the cartridge and any program in memory should be present on return.

There are cases however, where this is not true. In particular, the DOS 3 menu functions INIT DISK, COPY/APPEND, DUPLICATE, and ACCESS DOS 2 and HELP require more space than normally can be held in the MEM.SAV file. Therefore, if

you expect to be using these functions, the MEM.SAV is not able to save your entire program in memory and still allow DOS 3 to perform the function.

Therefore, you should always SAVE your program currently in memory using the BASIC SAVE or LIST instruction before typing DOS to perform one of these more complex DOS 3 functions. Basically, any function which has a file on the master diskette with the extension UTL needs more space, and if it is used, makes the MEM.SAV information invalid.

Whether you elect to change from MEM.SAV active to inactive or vice versa, the prompt you see when you type RETURN is:

**Updating KCP.SYS ...**

If MEM.SAV was active, the command response is then to erase the file named MEM.SAV from the disk. It also removes the dot from this item in the command area.

One final note ... if you have the MEM.SAV function active, and you have exited BASIC by typing: DOS

In this case there are three things you should not do if you wish the MEM.SAV function to restore your program to the memory on the return to the cartridge:
a. Do not perform one of the complex functions mentioned above,
b. Do not ERASE the file named MEM.SAV in disk unit 1,
c. Do not issue the To-Cartridge function until the same disk, which was in unit 1 at the time you exited to DOS, has again been placed in unit 1. This assures that the same MEM.SAV file, written at the time you exited to DOS, can be restored to the memory again.

**IF YOU NEED HELP**
While the computer is asking you for information, you may press the **inverse video** key or the **HELP** key to see the following summary of the MEM.SAV entry instructions:

**MEM SAVE HELP SCREEN 1 OF 1**

1. **Press M to select MEM.SAV option. When MEM.SAV is active, you can use a part of memory usually used by DOS.**
   **When using DOS with MEM.SAV on, the contents of lower memory are saved to disk in MEM.SAV file. When MEM.SAV is off, DOS may "overwrite" some of your data in memory.**
2. **Use MEM.SAV (Y/N)? Press Y to turn on MEM.SAV Press N to turn off MEM. SAV C/U. Press RETURN to leave MEM.SAV as is.**
3. **Check DOS menu: When MEM.SAV on, the name has a dot: Mem.Sav**
   **Press P to print this screen**
   **Press RETURN for more help**

After you have read the Help information, you have to press ESC to return to the main DOS 3 menu. Then you may restart the MEM.SAV function.

# HOW TO USE THE PROTECT FUNCTION

This function is selected from the main DOS 3 menu by pressing the letter P. It is used to modify one or more entries in the file index. Its purpose is to mark these entries as permanent files. Such files are not to be erased by the system using the erase utility, nor are they to be replaced by any file of the same name, or have anything else appended to them. Thus the name PROTECT.

When this function is selected, DOS 3 highlights the name in the menu area, and completes the word opposite the question:

**Selection? Protect**

Then DOS 3 asks:

**Filespec?**

The default reply, if you press **RETURN** here is:

**Filespec? D1:*.***

which means to protect all files on drive 1. You may edit this response if you wish, before pressing **RETURN**.
If you give a specific filespec, such as:

**D1:COPY.UTL**

Protection is added only to that file name on the source disk. However, if you specify any wildcards, you receive, as the next question:

**Protect all specified files (Y/N)#N?**

Here, the default reply, No, if you press **RETURN**, directs the system to ask you whether or not to protect each file it finds, in turn, matching the wildcard filespec you specify. If you change this reply to Y, for Yes, it does not ask any more questions and proceeds to protect all matching filespecs, reporting each as it is done.
   Note that the disk you are trying to "mark" the protection on must not itself be write-protected. This would prevent the updating of the file index and cause an error. It is not an error to add protection to a filename which is already protected.

**IF YOU NEED HELP**
While the computer is asking you for information, you may press the inverse video key or the **HELP** key to see the following summary of the Protect entry instructions:

**PROTECT HELP SCREEN 1 OF 1**
1. **Press P to write-protect files. Protected files cannot be written to, erased, renamed, or appended.**
2. **Filespec? Specify files to protect. Wildcards can be used.**
3. **Protect all specified files (Y/N)? Press Y to protect all files named by filespec. Press N or RETURN to protect files one at a time.**
4. **If protecting files one at a time, DOS shows you each filename and gives you a Y/N option. Check File Index. Protected files have (*) in front of filename.**

**Press P to print this screen**
**Press RETURN for more help**
**Press ESC for Help Menu**
After you have read the Help information, you have to press **ESC** to return to the main DOS 3 menu. Then you may restart the Protect function.

# HOW TO USE THE RENAME FUNCTION

The purpose of the Rename function is to change the name of one or more files on the disk. It is normally used on single files.
   The Rename function is entered by using the first letter R from the main DOS 3 menu. On entering this function, DOS 3 completes the word Rename opposite the word Selection on the command line. It also highlights this selection in the menu area. Now it asks the question:

**Old filespec?**

To ask which file or files you wish to rename. Wildcards may be used in the
filespec.

## FOR THE FIRST-TIME USER

This set of instructions provides a new file on the disk which you may then
Rename with this function. Note that this instruction sequence can only be done
with a disk that is NOT write-protected. This means you cannot perform this func-
tion on your ATARI DOS 3 Master Diskette, but you can use a copy of the master
which you already have made as a backup disk. Just be certain that the disk you
use is not write-protected.

1. From the main DOS 3 menu, select function F, File Index function.
2. When the computer asks
   **Filespec?**
   Press RETURN twice. This makes the default appear as
   **Filespec? D1:*.***
   and tells the computer to accept the default.
3. When the computer asks
   **Display Device?**
   Type D1:FILES
   and press RETURN
4. The file index appears on the screen and is stored in the disk file named FILES
   on disk unit 1.
5. Confirm that this name is present on the disk by performing the file index func-
   tion again. This time, select function F, as before, but press RETURN four times
   to accept all of the defaults. Now the file index appears on the screen and in-
   cludes the new name called FILES.
6. Now you can select the Rename function.

## WHEN DOS 3 ASKS ABOUT OLD FILESPEC

If you press RETURN, it shows the default filepec of D1:*.*. If you again press
RETURN to accept the default, it tells DOS 3 you want to either Rename all files
on the disk, or you can have DOS 3 present to you all of the file names, one at a
time, asking you for each if this file is to be renamed. Note that this second func-
tion is the most likely. No two files can have the same name, so once the correct
one is found, you won't want to rename others.

If you already know the name of the file you wish to erase, type it as the
filespec, example:
**D1:FILES**
or
**D:FILES**
or
**FILES**

all of which are equivalent. This is because D alone assumes the default D1.
Likewise the name alone assumes the default of D1. You may also specify
wildcards, in place of the D1:*.* you could specify:
**FI***

Here DOS 3 would present to you for decision, all files on drive 1 which began with
FI and ended in any manner.

## WHEN DOS 3 ASKS ABOUT NEW FILESPEC

The next question is:
**New Filespec? D1:**

48

Note that there is no default file name. You must provide a new name which DOS 3 uses for this file. For example:

**New filespec? D1:FILEINDX**

The specification stands for file index. Note that DOS 3 provides the device portion of the new filespec for you (the device portion may not be edited).

## HOW DOS 3 REPORTS PROGRESS

If you press **RETURN** at this point, DOS 3 searches the source device. If it matches the old filespec, it presents you with the question:

**Rename D1:FILES to FILEINDX (Y/N)? N**

In this case, you must change the N to a Y before pressing **RETURN** to actually rename the file. If you just press the **RETURN** at this point, you get the message, in the status line

**Job completed. 0 files renamed**

If you changed the N to a Y, then DOS 3 reports

**Renaming D1:FILES to FILEINDX**

and the status line will report

**D1:FILES Renamed**

You may, if you wish, specify wildcards in the selection of the old filespec, such as:

**Old Filespec? FI***

In this case, DOS 3 asks:

**Rename all specified files (Y/N)? N**
Note that the default N is already present. If you press **RETURN**, it means that DOS 3 is to search for all matches, and present each matching filename for your decision.

If you change the N to a Y, for Yes, it means just go ahead and Rename all files matching the filespec. In other words, don't ask questions, just do it. Normally, only one file is to be renamed.

In fact, this function halts, indicating an Error 174 - Duplicate Filename if you try to give the same name to more than one file. DOS 3 does not know which file you really want to continue having this name, so it refuses to create a problem in this manner.

Once all matching files have been found and questioned, bypassed or renamed, DOS 3 reports completion status on the status line of the main menu as:

**Job Completed. n files Renamed**

In this case, the n represents the total number of files actually renamed.

## IF YOU NEED HELP

While the computer is asking you for information, you may press the **inverse video** key or the **HELP** key to see the following summary of the RENAME entry instructions:

## RENAME HELP SCREEN 1 OF 1

1. **Press R to change the filename or extender of one or more files.**
2. **Old Filespec? Enter filespec to be renamed. Wildcards can be used.**
3. **New Filespec? Enter the new filespec. DOS will not allow duplicate filenames on a disk.**
4. **Rename all specified files (Y/N)? Press Y to rename all files in old filespec. Press N or RETURN to rename one at a time.**

**5. If renaming one at a time, DOS gives you a Y/N option on each.**

**Press P to print this screen**
**Press RETURN for more help**
**Press ESC for Help Menu**
After you have read the Help information, you have to press **ESC** to return to the main DOS 3 menu. Then you may restart the RENAME function.

# HOW TO USE THE SAVE FUNCTION

This function is called from the main DOS 3 menu by typing S. This function is generally used by the advanced user who has a need to save machine language memory areas for future use. As much as 64K of memory can be saved using a single command. However, only what might be called user-memory should be saved. This user-memory is defined in the memory map which is in the appendix of this manual, and includes a description of memory areas reserved for the system hardware, for the normal system software, and now used for DOS 3 as well. The rest of the memory, designated as user area, may be saved if desired.

When the S is received by DOS 3, it makes the selection line appear as follows:
**Selection? Save**

Then it highlights the SAVE selection in the menu, and it asks:
**Filespec?**

To this question, the user must respond with a filespec, consisting of a device and filename, such as:
**D1:NAME**
or
**D:NAME**
or
**NAME**

All of these examples save a file named"NAME"on disk drive 1. Drive 1 is the default in each case, but for this command, some input is required since there is no default for the name of the file to be saved. If you wish to Save a memory area onto drive 2, then the filename must begin with: D2:

Or for drive 3:
D3: and so on.

The next question asked is:
**Start Addr (HEX)?**

This command asks where, in the current memory, the hexadecimal (base 16) address is from which the data is to be saved. The allowable addresses to specify here are from hex 0000 to hex FFFF, followed by a **RETURN**. But as noted above, not all address ranges are practical to use. Typically a user saves certain memory spaces which he has loaded with machine language programs, somewhere in the region hex 2000 to about hex BFFF. (Below this region DOS resides, above this region, the system hardware and ROM reside.)

The next question is:
**End Addr (HEX)?**

This DOS 3 response asks for the entry of up to four hexadecimal digits in the range 0000 through FFFF, followed by a **RETURN**.

The ending address must be greater than or equal to the start address. If not, it generates an Error 6 - Invalid End Address and refuses to perform the command.

The next question is:

**Optional Init Addr (HEX)?**

This asks for an address of a maximum of four hex digits from 0000 to FFFF, then expects a **RETURN**. This address is recorded along with the memory data. When this file is again brought into the memory system using the DOS 3 Load command, if this INIT address was specified, the computer performs a JSR (jump to subroutine) to this address immediately following the Load. The purpose of this is to Initialize something, perhaps to allow proper running of the rest of the program (see also Run address below). This does not necessarily mean that the program begins immediately after Load. In essence, if the INIT program ends its task with an RTS (return from subroutine), control is returned to DOS 3.

This INIT address need NOT be within range of the memory space which was stored on the disk. In fact, often a user might want to bring in a memory segment, then use one of the system routines to reinitialize the screen or some other thing as the INIT routine. This feature forces a JSR to whatever address the user specifies, if any.

The next question is:

**Optional RUN addr (HEX)?**

It asks the user to specify up to four hex digits from 0000 to FFFF which is to be the RUN address. After the computer returns control to DOS 3 following the execution of the INIT function (if any), DOS 3 can, if directed to do so, give up control of the system by executing a JSR (direct JUMP) to this specified RUN address.

An example is as follows:

| | |
|---|---|
| **Start Addr (HEX)? 3000** | **RETURN** |
| **End Addr (HEX)? 3200** | **RETURN** |
| **Optional INIT addr (HEX)?** | **RETURN** |
| **Optional RUN addr (HEX)? E477** | **RETURN** |

The address E477 is used here as an example only. It is the cold start system reset address. It would not necessarily be logical to bring in a memory segment, then to tell the system to go through a cold start reset. However, it may be used to demonstrate that the RUN address does indeed cause a jump to the address specified. For reloading instructions, see the section on the DOS 3 Load function.

**IF YOU NEED HELP**
While the computer is asking you for information, you may press the **inverse** video key or the **HELP** key to see the following summary of the SAVE entry instructions:

**SAVE HELP SCREEN 1 OF 1**

1. **Press S to write a selected memory area onto disk. Save works ONLY with binary format programs.**
2. **Filename? Enter a unique filename. Wildcards are invalid input.**
3. **Start addr?/End addr? Enter start and end addresses in hex.**
4. **Optional INIT addr?/Optional RUN addr? Enter address of init program. Enter run address to load-and-go. Press RETURN to bypass init and run options.**

**Press P to print this screen**
**Press RETURN for more help**
**Press ESC for Help Menu**

After you have read the Help information, you have to press **ESC** twice to return to the main DOS 3 menu. Then you may restart the SAVE function.

# HOW TO USE THE TO-CARTRIDGE FUNCTION

This function is selected from the main menu by the letter T. Its purpose is to exit DOS, and to enter the cartridge. To make this command function, there must be a cartridge present, and you must have entered DOS 3 from the cartridge. To exit the cartridge, type: DOS

The main system menu appears as usual, with the command line specifying:
**Selection? To Cartridge**

If you specify a T and there is no cartridge present, the selection menu spells out your selection and the error line specifies an Error 4 -no cartridge.

If there is a cartridge present, it is initialized, either in the WARM start mode or the COLD start mode. In a COLD start, all programs the cartridge may have placed in memory are erased. If WARM start mode, the system is restored to its original status before DOS 3 was called from that language.

The primary difference lies in whether DOS 3 has run any of its "utilities" or not. The utilities are selected by the options Init, Copy, Duplicate, Help, or Access DOS 2. If none of these is used, the WARM start function can be used for the cartridge.

There are other considerations to COLD vs WARM start. The advanced user is directed to the ATARI Operating Systems Users Manual for more information. Also, other information may be found in the section titled "How to use the Mem.Sav Function."

**IF YOU NEED HELP**
While the computer is asking you for information, you may press the inverse video key or the HELP key to see the following summary of the To-Cartridge instructions:
**TO-CARTRIDGE HELP SCREEN 1 of 1**
1. **Press T to select function. To-Cartridge passes control from DOS to inserted cartridge.**
   (a) **If BASIC cart. is inserted, the screen displays a READY prompt.**
   (b) **If ASSEMBLER EDITOR is inserted, screen displays an EDIT prompt.**
   (c) **If you have not inserted cart., screen displays NO CARTRIDGE.**
2. **If you get message (c), press ESC to return to DOS menu.**

   **Press P to print this screen**
   **Press RETURN for more help**

After you have read the Help information, you have to press ESC to return to the main DOS 3 menu. Then you may restart the To-Cartridge function.

# HOW TO USE THE UNPROTECT FUNCTION

This function is selected from the main DOS 3 menu by pressing the letter U. It is used to modify one or more entries in the file index. Its purpose is to delete the protection mark placed on a filename by the Protect Function. This allows such files to be erased by the system either using the erase utility, or to be replaced by any file of the same name, or to have anything else appended to them. Thus the name UNPROTECT.

When this function is selected, DOS 3 highlights the name in the menu area, and completes the word opposite the question:
**Selection? Unprotect**

Then DOS 3 asks:
**Filespec?**

The default reply, if you press RETURN here is:

**Filespec? D1:*.***

This means remove the protection from all files on drive 1. You may edit this
response if you wish, before pressing RETURN. If you give a very specific filespec,
such as:

**D1:COPY.UTL**

In this case, protection is removed only from that file name on the source disk;
however, if you specify any wildcards, you receive, as the next question:

**Unprotect all specified files (Y/N) N?**

Here, the default reply, No, if you press RETURN, directs the system to ask you
whether or not to unprotect each file it finds, in turn, matching the wildcard
filespec you specify. If you change this reply to Y, for Yes, it asks no more ques-
tions and proceeds to remove protection from all matching filespecs, reporting
each as it is done.

    Note that the disk you are trying to unprotect must not itself be write protected.
This would prevent the updating of the file index and cause an error. Note also that
it is not an error to remove protection from an unprotected filename.

**IF YOU NEED HELP**

While the computer is asking you for information, you may press the **inverse video**
key or the **HELP** key to see the following summary of the Unprotect entry instruc-
tions.

**UNPROTECT HELP SCREEN 1 OF 1**

**1. Press U to select this function. UNPROTECT erases the protected status of
one or more files.**

**2. Filespec? Specify files to unprotect. Wildcards can be used.**

**3. Unprotect all specified files (Y/N)? Press Y to unprotect all files named by
filespec. Press N or RETURN to unprotect one by one.**

**4. In unprotecting files one at a time, DOS shows you each filename and gives
you a Y/N option. Check File Index. Unprotected files lose the (*) in front of
filename.**

**Press P to print this screen**
**Press RETURN for more help**
**Press ESC for Help Menu**

After you have read the Help information, you must press **ESC** to return to the
main DOS 3 menu. Then you may restart the Unprotect function.

# HOW TO USE THE X-USER-DEFINED FUNCTION

This function is for the advanced user, in connection with a machine language pro-
gram which has been Saved using the DOS 3 SAVE function. It is called by typing
the letter X from the main DOS 3 menu.

    On selecting the X-function, DOS 3 completes the phrase X-user-defined op-
posite the word Selection, and highlights this selection in the menu area. Then it
asks:

**External command?**

The external command is the name of a file with the three letter extension".CMD".

You must have previously Saved a file with a name like this in order to use this command. For example, if you had saved a file on the disk using the DOS 3 SAVE command, titled:

**MYOWN.CMD**

And then when DOS 3 asks:
**External command?**

You specify:
**MYOWN**

And press RETURN.

DOS 3 then responds:
**Loading D1:MYOWN.CMD ...**

And automatically executes it once it has been loaded.

This command is similar to the DOS 3 LOAD command. The difference is that for each file matching the filespec, DOS 3 asks if it should execute the file once it has been loaded. Then it goes on to try to match other file names to the filespec, asks if it should load each, then asks if it should execute each.

If you do not specify any file name when it asks about the external command, pressing RETURN results in DOS 3 searching the disk and presenting to you one at a time each file name which has the .CMD extension. On the following line then, it presents the prompt:

**Execute this command file (Y/N)? N**

Note the default reply, No, is already present. If you just press RETURN at this point, it continues the search, presenting each CMD file as it finds it. When you find the one you want to RUN, then change the N to a Y before pressing RETURN.

This has the same effect as using the DOS 3 LOAD command and requesting that this file IS to be run.

When it runs out of entries in the directory with the CMD as an extension, it reports, on the information line:

**No more matches**

Once a file has been loaded in this manner, because it will auto-execute, control is given to the file and not to DOS 3. To return control to DOS 3 (providing that the file does not interfere with memory in use by DOS 3), control may be returned to DOS by executing an RTS at the end of the file execution. Thus, these can truly be user-defined extensions to the DOS structure.

**IF YOU NEED HELP**
While the computer is asking you for information, you may press the inverse video key or the HELP key to see the following summary of the X-User-Defined entry instructions:

**User-Defined Help Screen 1 of 1**
1. **Press X to select a program you have written in assembly language and stored with SAVE.**
2. **After pressing X, you can (a) type in filespec and press RETURN or (b) press RETURN and let DOS show filenames on D1: with CMD extenders. If method (b) is used, press X with your program disk in drive 1.**
**After DOS loads in the command file, you may remove your program disk.**
**Press P to print this screen**
**Press RETURN for more help**
After you have read the Help information, you have to press ESC to return to the main DOS 3 menu. Then you may restart the X-user-defined function.

# MORE USER INFORMATION 5

## BASIC COMMANDS USED WITH DOS

Before describing the BASIC commands used with DOS 3, you need to know how the commands act on programs being stored and retrieved. The following paragraphs explain the two types of files that can hold BASIC programs.

## TOKENIZED AND UNTOKENIZED FILES

The first type of file, called "untokenized," contains standard ATASCII text characters so it looks like a printout of a BASIC program. These programs do not retain their symbol tables each time they are loaded and saved. The symbol table associates the variable name with the memory location where the values for that variable are stored. To store and retrieve a file in its untokenized form, you use the LIST and ENTER commands.

The second type, called a "tokenized" file, is a condensed version of a BASIC program. It has one-byte "tokens" instead of the ATASCII characters to represent the BASIC commands.

Tokenized programs are moved back and forth between the disk drive and the computer console by SAVE and LOAD commands. Tokenized versions of a file are generally shorter than untokenized versions. For this reason, many programmers prefer to store their final programs in the tokenized form because they load faster and use less disk space. A tokenized version retains its symbol table from retrieval to retrieval.

### LOAD (LO.)

**Format:**    LOAD filespec
**Example:**  LOAD "D1: DOSDEMO.BAS"
             RETURN

This command is used to load a file from a particular diskette in a disk drive into the user program RAM area. To use this command to load a file called DOSDEMO.BAS, the file (DOSDEMO.BAS) must have been previously saved using the BASIC command, SAVE. This command only loads a tokenized version of a program.

This command can also be used in "chaining" programs. If you have a program that is too big to run in your available RAM, you can use the LOAD command as the last line of the first program. Therefore, when the program encounters the LOAD statement, it automatically reads in the next part of the program from the diskette. However, the second program must be able to stand alone without depending on any variables or data in RAM from the first program. The loaded program does not execute until you type RUN **RETURN**, at which time the previous program and any variables are cleared (see RUN for another example). The following is an example of program chaining:

31100 REM Chain program
31200 LOAD "D1:CHAIN.BAS"

## SAVE (S.)

**Format:**    SAVE filespec
**Example:**  SAVE "D1:EXAMP2.BAS"
            RETURN

This command causes the computer system to save a program on diskette with the filespec name designated in the command. SAVE is the complement of LOAD and stores programs in tokenized form.

## LIST (L.)

**Format:**    LIST filespec ,lineno ,lineno device
**Example:**  LIST "D: DATFIL.LST"
            LIST "P:"
            LIST "P:", 10, 100

One use of the LIST command in BASIC is very similar to the SAVE command as it can take a program from user program RAM and store it onto a particular drive with any name you want to assign it (illustrated by the first example). However, the program is stored in standard ATASCII text and not as tokens. Differences in the formatting of data storage also allow LIST to be much more flexible than SAVE. As shown in the above format examples, you can specify a single device (e.g., P:, E:, C:, D:, D2:, etc.), or you can specify line numbers to be listed to a designated device (e.g., "P:", 100, 200).

If you do not specify a device after the LIST command, any line numbers you enter are displayed on the screen. The screen (E:) is always the default device for this command.

In summary, the principal difference between LIST and SAVE is that LIST moves standard ATASCII text to a number of different devices whereas SAVE can only save tokenized BASIC programs on a diskette.

Because files placed on the diskette with LIST are not tokenized, you may, if you wish, use the DOS 3 COPY command to copy from a LISTED file to the printer. See the COPY function for more information.

## ENTER (E.)

**Format:**    ENTER filespec
**Example:**  ENTER "D: LIST2.LST"

This command causes the computer to move a file on diskette with the referenced filespec into RAM. The program is entered in untokenized form and is interpreted as the data is received. ENTER, unlike LOAD, does not destroy a RAM-resident BASIC program, but merges the RAM-resident program and the disk file being loaded. If there are duplicate line numbers in the two programs, the line in the program being entered replaces the same line in the RAM-resident program.

## RUN

**Format:**    RUN filespec
**Example:**  RUN "D2: MYFILE.BAS"

This command causes the computer to LOAD and RUN the designated filespec. It is a combination of the two commands, LOAD and RUN. However, the RUN command can only be used with tokenized files. Therefore, you cannot execute a RUN "D2: LIST.LST"command.

To chain programs and cause a second segment of a file to load and run automatically, you can use a RUN "D: filespec"as the last line of the first segment. However, the second program must be able to stand alone without depending on any variables or data in RAM from the first program. Before running the first segment, make sure you have saved it on a diskette, as the RUN statement wipes out your RAM-resident first segment when the second segment is loaded.

## INPUT/OUTPUT CONTROL BLOCKS

An I/O operation is controlled by an I/O Control Block (IOCB). An IOCB is a specification of the I/O operation, consisting of the type of I/O, the buffer length, the buffer address, and two more auxiliary control variables of which the second is usually 0. ATARI BASIC sets up eight IOCBs and dedicates three to the following:

- IOCB #0 is used by BASIC for I/O to E: (the screen editor)
- IOCB #6 is used by BASIC for I/O to S: (the screen display controller)
- IOCB #7 is used by BASIC for LPRINT, CLOAD, and SAVE commands.

IOCB #1 through #5 can be used freely, but the dedicated IOCBs should be avoided unless a program does not make use of one of the dedicated uses mentioned above. IOCB #0 can never be opened or closed from a BASIC program.

## IOCBs WITH INPUT/OUTPUT COMMANDS

Each input/output command must have an IOCB associated with it. The I/O commands that can be used in connection with DOS 3 are the following:
**OPEN/CLOSE**
**INPUT/PRINT**
**PUT/GET**
**STATUS**
**XIO**

## USING THE OPEN/CLOSE COMMANDS

### OPEN (O.)

**Format:**    OPEN #iocb, aexp1, aexp2, filespec
**Example:**   100 OPEN #2, 8, 0, "D1: ATARI800.BAS"

The OPEN statement links a specific IOCB to the appropriate device handler, initializes any CIO-related control variables (see Glossary), and passes any device-specific options to the device handler. The parameters in this statement are defined in the following:

**#**        Mandatory character entered by user.
**iocb**     A number between 1 and 7 that refers to a device or file.
**aexp1**    Number that determines the type of operation to be performed.

         Code 4  = input operation; positions file pointer to start of file.
            6   = disk directory input operation.
            8   = output operation; positions file pointer to start of file.
            9   = end-of-file append operation; positions file pointer to end of file. Code 9 allows program input from screen editor without user pressing **RETURN**.

12 = input and output operation; positions file pointer to start of file, not possible to write past EOF.

13 = Input and Output operation; positions file pointer to EOF; writing past EOF is allowed.

**aexp2**    Device-dependent auxiliary code. An 83 (ASCII S) in this position causes the ATARI 820™ Printer to print sideways; otherwise it is always 0 (zero).

**filespec**    Specific file designation (see Section 1 for filespec definition).

In the example, OPEN #2, 8, 0,"D1: ATARI800.BAS", IOCB #2 is opened for output to a file on Drive 1 designated as ATARI800.BAS. If there is no file by that name in Drive 1, the DOS creates one. If a file by that name already exists, the OPEN statement destroys that file and creates a new one. If the IOCB has already been opened, the screen displays an ERROR-129 (IOCB Already Opened).

### CLOSE (CL.)

**Format:**    CLOSE #iocb
**Example:**    300 CLOSE #2

The CLOSE command releases the IOCB that had been previously opened for read/write operations. The number following the mandatory # must be the same as the IOCB reference number used in the OPEN statement (see example below). If the IOCB has already been opened to one device and an attempt is made to open the same IOCB to another device without closing, the first ERROR-129 displays on the screen. The same IOCB cannot be used for more than one device at a time. You do not get an error message if you close a file that has already been closed. The following is an example of opening and closing a file:

10 OPEN #1, 8, 0,"D: FIL.BAS"
20 CLOSE #1

Note: The END command closes all open files (except IOCB #0).

## USING THE INPUT/PRINT COMMANDS

### INPUT (I.)
**Format:**    , avar avar
            INPUT #iocb ; svar , svar ...
**Examples:**    100 INPUT #2; X, Y
            100 INPUT #2; N$

This command is used to request data (either numerical or string) from a specified device. INPUT is the complement of PRINT. When used without an #IOCB, the data is assumed to be from the default device (E:). INPUT uses record I/O (see PRINT).

The following is an example of an INPUT/PRINT program:

```
10 REM *CREATE DATA FILE*
20 REM *OPEN WITH 8 CREATES DATA FILE*
30 OPEN #1,8,0,"D:WRITE DAT"
40 DIM WRT$(60)
50 ?"ENTER A SENTENCE NOT MORE THAN 60 CHARACTERS."
60 INPUT WRT$
70 REM *WRITE DATA TO DISKETTE*
80 PRINT #1; WRT$
90 REM *CLOSE FILE*
100 CLOSE #1
110 REM *OPEN DATA FILE FOR READ*
120 REM *OPEN WITH 4 IS A READ ONLY*
130 OPEN #1,4,0,"D:WRITE.DAT"
140 REM *READ DATA FROM DISKETTE*
150 INPUT #1,WRT$
160 REM * PRINT DATA *
170 PRINT WRT$
180 REM *CLOSE DATA FILE*
190 CLOSE #1
```

In the example above, line 60 allows the user to type in data on the keyboard (default device). In Line 150, the INPUT statement reads the contents of the string from the opened file.

**PRINT (PR. or ?)**
**Format:**    PRINT #iocb ; [exp]... [exp]

**Examples:** 100 PRINT #2; X, Y
              100 PRINT #2; A$
              100 ? C$
              100 PRINT"X = ",X

This command writes an expression (whether string or arithmetic) to the opened device with the same IOCB reference number. If no IOCB number is specified, the system writes the expression to the screen, which is the default device. If the information is directed to a device that is not open, ERROR-133 displays on the screen. PRINT performs what is called record I/O. Records are sets of bytes separated by end-of-line characters (9B Hex). The size of a record is arbitrary. Record size can be determined by the length of a string printed to a diskette file or the format of an arithmetic variable. It can also be the length of a string of characters entered from the keyboard and terminated by **RETURN**.

The INPUT statement cannot (generally) read a record that is longer than 110 characters in length. If you PRINT a record to the disk that you later want to INPUT, it is best to limit the size of the PRINTED records to 110 characters or less.

# DIRECT ACCESSING WITH THE NOTE/POINT COMMANDS

## NOTE (NO.)

**Format:**   NOTE #iocb,avar,avar
**Example:**   NOTE #2,A,B

Files are created sequentially and are normally accessed from beginning to end. In other words, when you create a data file, each record may have been placed in the file by some form of PRINT statement. As an example, there may be a file named "STUDENT.DAT" in which there are 200 "records" (specific groupings of data). If you wanted to read only record numbers 25, 120 and 190, the normal sequential access method would take a long time. The sequential method means to have an INPUT statement in a loop reading each one of the records in sequence, ignoring each one until it counts up to the one you really wanted to read.

   The situation would be even worse if you needed the records in a truly random manner. The sequential example above would have required 190 disk reads before you would be able to get all the data you wanted. If, instead of the above sequence you wanted them in the order 190, 120, 25, you might have been required to reset the file pointer to the beginning EACH time, then to count up to the record you wanted. This would required 190 + 120 + 25 or 335 disk reads.

   To eliminate such delays in nonsequential file access, the NOTE command is used to "make a note" of where in the file each individual record is stored. Later then, using the POINT command, the file pointer can be placed directly on the desired record, allowing immediate access by a single read instead of the multiple read required in the example described above. This is called random file access.

   The following program sample below, called NOTETEST, demonstrates the use of the NOTE command. DOS 3 treats all files as a sequence of data bytes. This sequence begins with the first byte, labeled as byte "0", with each subsequent byte at position 1,2, 3 etc. The value of the first "avar" (in the example program, this is the variable X) represents the value of the current pointer to the file. That is, where in the file the next byte is be stored. This means that in a file freshly opened, this value is 0.

   The second "avar" (in the sample program, Y), is meaningless in DOS 3 but MUST BE PRESENT. This is because ATARI BASIC was written for compatibility with DOS 2 initially. See the section called "Differences between DOS 3 and DOS 2" for more information. When you RUN the sample program NOTETEST, you can type in a number of lines, each with 40 characters or less. When you press RETURN with no characters on a line, the files are closed and the program ends. This builds a pair of files for the POINTEST program. The comments in the program tell you how it works.

```
10 REM NOTETEST DEMO
20 REM CREATES A POINTER FILE
30 REM FOR DOS 3 RANDOM ACCESS
40 DIM A$(40):COUNT = 0
50 OPEN #1,8,0,"D:DATFIL.DAT"
60 OPEN #2,8,0,"D:POINTS.DAT"
70 REM INPUT 40 OR LESS CHARACTERS
80 INPUT A$
90 REM IF RETURN ONLY, THEN STOP
100 IF LEN(A$) = 0 THEN 250
```

```
110 NOTE #1,X,Y:REM ONLY X IS SIGNIFICANT
120 REM STORE LINE OF DATA
130 PRINT #1;A$
135 COUNT = COUNT + 1:IF COUNT = 100 THEN GOTO 250
140 REM STORE POINTER TO BEGINNING
150 REM LINE OF DATA
160 REM DOS 3 REQUIRES ONLY ONE
170 REM PARAMETER
180 PRINT #2,X
190 REM DISPLAY OFFSET TO DATA
200 REM LINES ON SCREEN
210 PRINT "OFFSET = ";X
220 REM NOW INPUT NEXT RECORD
230 GOTO 80
240 REM INDICATE END OF FILE
250 PRINT #2,-1
260 END
```

## POINT(P.)

**Format:**    POINT #iocb, avar, avar
**Example:**   100 POINT #2, A, B

POINT is the complement of NOTE. This command sets the file pointer to an arbitrary value determined by the arithmetic variables. POINT is used when reading or writing specified file locations (sector and byte) into RAM or to disk. The first arithmetic variable specifies (points to) the next byte number into which the next byte is read or written. If you point out of an opened file, you get a File Number Mismatch error message. The following program listing and sample run that follow it contain an example of the POINT command to read data created by the program shown as the example for the NOTE command.

When run, this program allows the user to choose a record to display.

```
10 REM POINTEST DEMO
20 REM THIS PROGRAM READS
30 REM THE FILES CREATED
40 REM BY POINTEST AND ACCESSES
50 REM THE RECORDS RANDOMLY
60 DIM A$(40)
70 DIM X(100):Y = 0:LAST = 0
80 REM OPEN DATA FILE
90 OPEN #1,4,0,"D:DATFIL.DAT"
100 REM OPEN POINTER FILE
110 OPEN #2,4,0,"D:POINTS.DAT"
120 REM READ POINTERS INTO AN ARRAY
140 INPUT #2,P:IF P < > -1 THEN LAST = LAST + 1:X(LAST) = P:GOTO 140
180 REM INPUT RECORD NUMBER TO DISPLAY
185 TRAP 320
190 PRINT "ENTER RECORD NUMBER TO DISPLAY"
200 INPUT R
```

```
205 IF R < 1 THEN GOTO 280:REM END PROGRAM
210 IF R > LAST THEN PRINT "RECORD DOES NOT EXIST":GOTO 190
220 REM ACCESS RECORD RANDOMLY
230 POINT #1,X(R),Y
240 INPUT #1;A$
250 PRINT A$
260 GOTO 180
280 PRINT "END OF EXECUTION"
290 CLOSE #1:CLOSE #2
300 STOP
320 TRAP 40000:GOTO 185:REM IF INPUT ERROR CONTINUE INPUT
```

# USING THE PUT/GET COMMANDS

PUT (PU.)

Format:     PUT #iocb, aexp

**Example:**   100 PUT #6, ASC ("A")

The PUT command writes a single byte (value from 0-255) to the device specified by the IOCB reference number. In Figure 5-1 the PUT command is used to write each number you type into an array dimensioned as B(50). You can enter up to 50 numbers, each of which should be less than 256. This command is used to create data files or to append data to an existing file.

The sample program shown below is split into two parts. The first part demonstrates PUT, and is shown here. The second part demonstrates GET and is contained under the GET heading. In the example for GET, rather than sense an end-of-file (TRAP on error) to determine the end of the data, bytes 0 and 1 of the file itself have been used as a counter of the number of bytes in the file.

```
10 GRAPHICS 0:REM PUT/GET DEMO
20 DIM A(50),A$(10)
30 GRAPHICS 0:?"PUT AND GET TO DISK PR
OGRAM EXAMPLE":?
40 ? "Is this to be a READ or a WRITE?":
INPUT A$:?
50 IF A$ = "READ";THEN 170
60 IF A$ < > "WRITE" THEN PRINT "?":GOTO 40

70 REM WRITE ROUTINE
80 OPEN #18,0,"D1:EXAMPL1.DAT"
90 ? "Enter a number less than 256":INPUT
X
95 REM **WRITE NUMBER TO FILE**
100 PUT #1,X
110 IF X = 0 THEN CLOSE #1:GOTO 130
120 GOTO 90
```

**Figure 5-1 Sample PUT Program**

**GET (GE.)**

**Format:**     GET #iocb, avar

**Example:**   100 GET #2, X

This command reads a single byte from the device specified by the IOCB reference number into the specified variable. The second part of the program example (Figure 5-2) illustrates the GET command. It allows you to retrieve each byte stored by the PUT command.

```
130 GRAPHICS 0:?:? "Read data in file n
ow?":INPUT A$?
140 IF A$ = "NO" THEN END
150 IF A$ < > "YES" THEN 130
160 REM READ OUT ROUTINE
170 OPEN ,4,0,"D1:EXAMPL1.DAT"
180 FOR E = 1 TO 50
185 REM **READ NUMBER(S) FROM FILE**
190 GET #2,G:A(E) = G
200 IF G = 0 THEN GOTO 230
210 PRINT "BYTE #",E;" = ";G
220 NEXT E
230 CLOSE #2
```

**Figure 5-2 Continuation of the PUT/GET Demo**

Note that INPUT/PRINT and GET/PUT are incompatible types of INPUT/OUTPUT. PRINT inserts end-of-line (EOL) characters between records and INPUT uses them to determine a record. GET and PUT merely write single bytes to a file without separating them with EOL. A file created by using PUT statements looks like one large record unless you have placed an EOL (9B Hex) character into the file.

After you have typed the sample PUT/GET program, type RUN RETURN.

When you run the program shown in Figure 5-2 it prints the numbers entered from the keyboard together with the byte numbers in which each was stored.

After you type the program, type RUN RETURN using number entries 2, 5, 67, 54, 68.

```
BYTE #1 = 2
BYTE #2 = 5
BYTE #3 = 67
BYTE #4 = 54
BYTE #5 = 68
```

**Figure 5-3. Sample Run of The PUT/GET Program**

# USING THE STATUS COMMAND

### STATUS (ST.)

**Format:**  STATUS #IOCB, avar
**Example:**  100 STATUS #5, ERROR

The STATUS command is used to determine the condition (state) of a file. This command is a CIO command and checks for several ways an error occurs. The first set of possible errors it checks for is as follows:

System buffer available? If no, then ERROR-161
Legal device number?   If no, then ERROR-160
Legal filename?      If no, then ERROR-165
File on diskette?     If no, then ERROR-170
File protected?      If yes, then ERROR-167

You can also identify all I/O serial bus errors with a STATUS command. These are as follows:

| | |
|---|---|
| Device timeout | ERROR-138 |
| Device not acknowledged | ERROR-139 |
| Serial bus error | ERROR-140 |
| Serial bus data frame overrun | ERROR-141 |
| Serial bus checksum error | ERROR-142 |
| Device done | ERROR-144 |

To use this command, you must open the file as an input-only file, then close the file. Only then can you issue a STATUS command.

Figure 5-4 allows you to check the status of your disk drive with a TRAP statement. Before running the program, turn off your disk drive.

```
10 GRAPHICS 0:REM TRAP/STATUS DEMO
20 DIM A(50),A$(10),D$(1)
30 GRAPHICS 0:?"PUT AND GET TO DISK PR
OGRAM EXAMPLE":?
40 ? "Is this to be a READ or a WRITE?":
INPUT A$:?
50 IF A$ = "READ" THEN 160
60 IF A$ < > "WRITE" THEN PRINT "?":GOTO 40

70 REM WRITE ROUTINE
80 TRAP 400:OPEN #1,8,0,"D1:EXAMPL1.DAT"

90 ? "Enter a number less than 256":INPU
T X
100 PUT #1,X
110 IF X = 0 THEN CLOSE #1:GOTO 130
120 GOTO 90
130 GRAPHICS 0:? :? "Read data in file n
ow?":INPUT A$:?
140 IF A$ = "NO" THEN END
150 IF A$   "YES" THEN 130
160 REM READ OUT ROUTINE
170 TRAP 400:OPEN #1,4,0,"D1:EXAMPL1.DAT
"
180 FOR E = 1 TO 50
190 GET #1,G:A(E) = G
200 IF G = 0 THEN GOTO 230
210 PRINT "BYTE #";E;" = ";G
220 NEXT E
230 CLOSE #1
240 END

400 TRAP 40000:STATUS #1,ST:IF ST < > 138 A
ND ST < > 139 THEN PRINT "HELP":? ST:GOTO 4
30
410 ? "Is your disk drive turned on?"
420 ? "Type Y if you turned on the disk
drive.":INPUT D$
430 CLOSE #1:GOTO 40
```

**Figure 5-4 Sample Status Program**

## OTHER COMMAND NUMBERS USED BY DOS 3

The following command numbers are used internally to DOS 3. They could be specified as part of an XIO command. However they would be meaningless in that there is no place within the command to specify or to accept a variable which is normally passed to the standard form of the command. Therefore, these nonstandard forms cannot be used and the normal BASIC command must be used instead.

| Command # | TYPE | Equivalent BASIC Command |
|---|---|---|
| XIO 5 .... | Get record | INPUT#1,A,B,C or INPUT F$ |
| XIO 7 .... | Get character | GET#1,A |
| XIO 9 .... | Put record | PRINT#1,A,B,C or PRINT F$ |
| XIO 11 ... | Put character | PUT#1,A |
| XIO 37 ... | Point | POINT#1,X,Y |
| XIO 38 ... | Note | NOTE#1,X,Y |
| XIO 39 ... | Get directory entry | INPUT#1,FILE$ (see example) |
| XIO 40 ... | Rename a filespec | There is no functional equivalent. DOS 3 uses it internally to convert a wildcard filespec into a real one. |

## PROGRAM EXAMPLE OF XIO COMMAND USES

Figure 5-5 allows you to create a file for each month of the year into which you can enter the names and birthdays of your family and friends. The program uses XIO statements to create a file for each month, to lock and unlock each file as needed by the program, and to close the file when you are through with it.

Line 20 defines the disk file D: BIRTHDAY as FILE$. Then in Line 170, FILE$ is opened with a XIO statement for input. The XIO statement in Line 390 unlocks the proper file. The XIO statement in Line 400 creates the file and allows you to write to the file. The next XIO statement, in Line 430, closes the file and the next line's XIO statement locks the file to prevent it from being accidentally overwritten or erased.

```
5 GRAPHICS 0
10 DIM A$(5),D$(15),FILE$(20),DATE$(20),
MON$(20),ERR$(20),NAME$(40)
20 FILE$ = "D:BIRTHDAY."
30 ERR$ = "ERROR IN MONTH #"
100 GRAPHICS 0:? "PLEASE TYPE MONTH NUMB
ER (1-12)"
110 TRAP 100:INPUT MONTH
120 TSTEND = 0
130 MONTH = INT(MONTH)
140 IF MONTH  1 OR MONTH  12 THEN ? ERR$:G
OTO 100
145 GOSUB 1000 + MONTH
150 FILE$(12) = STR$(MONTH)
160 EOF = 0
170 TRAP 700:XIO 3,#2,4,0,FILE$
180 TRAP 600:FOR I = 0 TO 1 STEP 0
190 INPUT #2;NAME$
200 INPUT #2;DATE$
210 EOF = EOF + 1
220 IF EOF = 1 THEN ? "BIRTHDAYS IN ";MON$
;"ARE:":?
224 TEMP = LEN (NAME$)
225 NAME$(TEMP + 1) = " "
```

**Figure 5-5. Sample XIO Program**

```
226 NAME$(30) = " "
227 NAME$(TEMP + 2,30) = NAME$(TEMP + 1)
230 ? NAME$,DATE$
240 NEXT
299 REM MKE NEW ENTRIES IN BIRTHDAYS
300 ? "WOULD YOU LIKE TO MAKE A NEW BIRT
HDAY ENTRY":INPUT A$
310 IF A$ > <"YES" THEN GOTO 20
320 ? "PLEASE TYPE PERSONS NAME"
330 INPUT NAME$
340 ? "PLEASE TYPE PERSONS BIRTHDAY (MM-
DO-YY)"
350 INPUT DATE$
360 MONTH = INT(VAL(DATE$))
370 IF MONTH < 1OR MONTH > 12 THEN ? ERR$,D
ATE$:GOTO 300
380 FILE$(12) = STR$(MONTH)
390 TRAP 400:XIO 36,#3,0,0,FILE$:OPEN #
,9,0,FILE$:GOTO 410
400 CLOSE #2:XIO 3, #2,8,0,FILE$
410 PRINT #2;NAME$
420 PRINT #2;DATE$
430 XIO 12,#2,0,0,FILE$
440 XIO 35,#2,0,0,FILE$
450 GOTO 300
600 CLOSE #2:IF EOF = 0 THEN ? "NO BIRTHDA
YS IN ";MON$
610 MONTH = MONTH + 1
620 IF MONTH  12 THEN MONTH = 1
630 TSTEND = TSTEND + 1
640 IF TSTEND = 1 THEN GOTO 145
650 GOTO 300
700 EOF = 0:GOTO 600
1001 MON$ = "JANUARY":RETURN
1002 MON$ = "FEBRUARY":RETURN
1003 MON$ = "MARCH":RETURN
1004 MON$ = "APRIL":RETURN
1005 MON$ = "MAY":RETURN
1006 MON$ = "JUNE":RETURN
1007 MON$ = "JULY":RETURN
1008 MON$ = "AUGUST":RETURN
1009 MON$ = "SEPTEMBER":RETURN
1010 MON$ = "OCTOBER":RETURN
1011 MON$ = "NOVEMBER":RETURN
1012 MON$ = "DECEMBER":RETURN
3000 FILE$(12,12 + LEN(STR$(MONTH))) = STR$
MONTH):? FILE$
```

**Figure 5-5 Sample XIO Program(Cont.)**

When you run this program, enter a number from 1 to 12. The program checks to see whether or not there are any entries in the file. If there are none, the screen displays the message NO BIRTHDAYS IN followed by whatever month you selected. If you have made entries, the screen displays the names of the people and their birthdays for the month. In either event, the screen displays the names and birthdays for the month you selected and the succeeding month (as a failsafe feature so you do not forget an important birthday that comes at the first part of the next month). When you do not wish to see another file or make another birthday entry, type NO to each prompt message and the program ends.

# SUBSTITUTING THE XIO COMMAND FOR DOS MENU OPTIONS

### XIO (X.)

**Format:**   XIO cmdno, #iocb, aexp1, aexp2, filespec
**Example:**   100 XIO 3, #6, 4, 0,"D: TEST.BAS"

The XIO command is a general INPUT/OUTPUT statement used for special operations. It is used when you want to perform some of the functions that would otherwise be performed using the DOS menu selections. Note that XIO calls need filespecs. The various operations possible using XIO with DOS 3 are summarized below.
CMDNO (command number) is used to designate just which of the operations is to be performed.

| CMDNO OPERATION | EXAMPLE |
|---|---|

 3    OPEN                                      XIO 3, #1, 4, 0,"D:TEST.DAT"
      This command functions exactly like the OPEN command, explained
      earlier. It opens a file for read operations (parameter shown as a 4 above).
      Equivalent to the command
      OPEN #1,4,0,"D:TEST.DATA"

 12   CLOSE                                     XIO 12, #1, 0, 0,"D:TEST.DAT"
      This command is equivalent to the command CLOSE #2. Therefore this
      form of the close command is seldom used.

 32   RENAME                                    XIO 32, #1, 0, 0,"D:OLD,NEW"
      This command allows you to rename a file on the disk without calling the
      DOS 3 menu. The first part of the filespec is D:OLD which consists of the
      complete filespec for the old source file name which is to be changed.
      Other examples for this part are:
          D1:OLDNAME.BAS
          FORMER.NAM (assumes unit 1)

          The second part of the filestring is separated from the first by a
          comma. In this example, it is the name NEW. Note that when you
          use this form, you do NOT specify a unit number after the comma.
          The file is renamed on the disk on which the old filespec is made.

          The effect of the command is identical to that of calling the DOS 3
          main menu and executing the RENAME file command.
      If you use wildcards in the source filespec, the Rename occurs with the
      first file (starting at the beginning of the file index) which matches the
      filespec. If there are more than one file which matches this filespec, an
      error 174 occurs.

      Note that when an IOCB is selected for any one of these special DOS-
      from-BASIC operation, it must not already be in use for another kind of
      operation, such as an IOCB opened for a file read or write. For each of
      the commands shown below, the IOCB is used, then is free for further use
      after command completion. This is in contrast to the OPEN command
      which ties up the IOCB until the file is closed or the program ends (which
      automatically closes open files).

33  ERASE                                    XIO 33, #1, 0, 0,"D2:JUNK.BAS"
This command, used from BASIC, causes the deletion of the file matching
the filespec and has the same effect as calling the ERASE function from
the DOS 3 menu. If there is no file on the selected disk matching the
filespec, then an error 170 (file not found) is generated.

Wildcard may be used, such as:
    XIO 33,#1,0,0,"D2:J*.*"
which has the result of erasing all files that match the filespec.
34  VALIDATE FILENAME             XIO 34, #1,0,0, F$
This command can be used to find out if a specified name is of a valid
form. It would normally be followed by a STATUS command, as:

    STATUS #1,A

If the file name is not valid, the variable A is set to 130 (nonexistent
device), 160 (device number error), or 165 (filename error). If the file name
is valid, the status is set to 1. Wildcards may be used in the filespec.
35  PROTECT                              XIO 35, #1, 0, 0,"D:PRECIOUS"
36  UNPROTECT                         XIO 36, #1, 0, 0,"D:OKWRITE.FIL"
These commands are the exact equals of the PROTECT and UNPROTECT
function calls from the main DOS 3 menu. The result is the same.
41  LOAD A FILE                        XIO 41,#1,0,0,"D2:LOAD.FIL"
This has the same effect as calling the LOAD function from the main DOS
3 menu. The INIT address, if any, is executed as is the RUN address. If
the file which is loaded does not affect any of the space which is being
used by BASIC, an RTS at the end of the INIT part returns control to
BASIC. This function is for the advanced user.
253  FORMAT ANY DISK
For the 810, this command is:
    XIO 253,#1,33,87,"D2:"
    (or whichever unit to be done)

    For the 1050 single density:
    XIO 253,#1,33,87,"D2:"(same as 810)

    For the 1050 double density:
    XIO 253,#1,33,127,"D2:"

    Both the 810, and the 1050-single density mode format 720 sectors
    on the disk. The command byte 87 above (aux2) represents the
    result of the formula (720-24)/8.

    The 1050, in double density mode is formatted with 1040 sectors.
    Thus, the aux2 command byte is (1040-24)/8, or 127.
254  FORMAT AN 810 DISK
    XIO 254,#1,0,0,"D2:"
In this case, the aux1, and aux2 bytes are ignored. The selected unit is
told to format a disk in the 810 mode.

# ACCESSING DAMAGED FILES

There are two types of damaged files:
1.  The disk directory entry, which contains the file name, directory pointer (points to the first sector of the file), and the number of sectors in the file.
2.  The file itself.

Should the DISK DIRECTORY entry be damaged, there is no way to access the file. If the DISK DIRECTORY entry was accidentally deleted, FMS reports ERROR 170 (File Not Found). If the number of blocks indicated in the DISK DIRECTORY entry do not coincide with the actual number of blocks in the file, FMS reports ERROR 177 (Disk Structure Damage). In this latter case, you may be able to retrieve a portion of the file by initiating the Get Byte program in Figure 5-6.

```
READY

10 OPEN #1,4,0,"D:FILE.1"
20 OPEN #2,8,0,"D:FILE.2"
25 TRAP 50
30 GET #1,A
40 PUT #2,B
45 GOTO 30
50 CLOSE #1
60 CLOSE #2
```

**Figure 5-6 Get Byte Program**

Let us say File 1 = the damaged file, and File 2 = the recovery file. Note: You can only read the sectors that fall BEFORE the damaged sector(s). All other sectors after the damage cannot be accessed. As a result, it would be best to COPY the good files on the damaged diskette to a new diskette to avoid any further problems.

If the file itself is damaged, you can also use the Get Byte program which will transfer each good sector from the damaged file into a recovery file.

# THE AUTORUN.SYS FILE

When an AUTORUN.SYS file exists on the diskette in drive 1, that file is automatically loaded into RAM and executed (if appropriate) every time you boot the system. This entire process is completed before control of the system is returned to you. The AUTORUN.SYS file can be data; it can also be object code that is loaded, but not executed; or, it could be object code that is loaded and then executed as soon as the load is complete.

Figure 5-7 illustrates the use of AUTORUN.SYS to boot up directly into DOS even if a cartridge is present. After execution, AUTORUN.SYS normally returns to the DOS initialization routine. If during your application, it does not return, or if you allow the use of SYSTEM RESET before the return, the system initialization must be completed before proceeding. This is done by modifying two Operating System storage locations: COLDST at address 244 (Hex) and BOOT at address 9 (Hex). COLDST should be cleared to 00 and BOOT is set to 01.

The program listed below sets these two locations to the proper value and then jumps indirectly to the start DOS vector.

If you do not have an Assembler Editor cartridge, you can create the equivalent file using BASIC POKE statements and then saving the Binary File in DOS. The list of decimal numbers to be entered is as follows:

| Decimal Address | Decimal Codes |
| --- | --- |
| 15000 | 162, 0 |
| 15002 | 142, 68, 2 |
| 15005 | 232 |
| 15006 | 134, 9 |
| 15008 | 108, 10, 0 |

When these codes have been entered in BASIC, type DOS **RETURN** to save the file using the SAVE selection from the DOS menu.

Notice that there is no number entered for the INIT parameter. If you turn off your computer and then turn it back on, your program should now boot up directly into DOS. To enter BASIC, simply use the To-Cartridge function or press **SYSTEM RESET**.

```
; Autorun Program
; Run DOS without going to cartridge.
COLDST = $244
BOOT = $09
DOSVEC = $0A
* = 3A98
                              (HEX CODE)
DOSGO     LDX #0              A2 00
          STX COLDST         8E 44 02
          INX                E8
          STX BOOT           86 09
          JMP (DOSVEC)       6C 0A 00
          * = $2E0           run address at 2E0
          . WORD DOSGO       98 3A
          . END
```

**Figure 5-7. An AUTORUN.SYS Example for the Advanced User**

Please refer to the use of the SAVE function for detailed instructions on SAVEing this program. The start, end, addresses to be specified are 3A98, 3AA2. For INIT address, simply press **RETURN**. For the RUN address, specify 3A98.

# ADDITIONAL INFORMATION ABOUT THE DISK DRIVE SYSTEM 6

## ATARI DISKETTES

ATARI Diskettes are thin mylar circular sheets covered with an oxide similar to that used on cassette tape. Each ATARI Diskette is 5 1/4 inches in diameter and each is sealed in a special black jacket designed to protect it from being bent, scratched, or contaminated.

Each disk drive requires the diskette created specifically to operate with that drive. The ATARI 810 Disk Drive is a single-density drive and the ATARI 1050 Disk Drive can be used as single or a double-density disk drive. The essential difference between double-density diskettes and single-density diskettes is that the double-density recording technique requires a higher quality recording surface so that it can store more data in the same space. Both types of diskettes are manufactured in the same way. The difference is that the double-density diskettes have been pretested to guarantee performance with the double-density recording techniques. A blank double-density diskette should work as well on a single-density disk drive, but you should not use a blank, single-density diskette on a double-density disk drive.

Once a diskette has been formatted for use with a single-density disk drive it cannot be used in a double-density disk drive unless you reformat the diskette on the double-density drive. The reverse is also true. If you have a system that uses both types of drives and diskettes, be sure that you clearly label each diskette to show the type of drive on which it was formatted.

## ATARI 810 DISK DRIVE

The ATARI 810 Disk Drive is a single drive with single-density recording capabilities. It uses standard 5 1/4-inch flexible diskettes: ATARI 810 Master Diskette II (CX8104), ATARI 810 Blank Diskettes (CX8100), and ATARI 810/815 Blank Diskettes (CX8202), each of which stores 90K (90 thousand) bytes. The ATARI 810 Disk Drive contains a built-in microprocessor which gives it an automatic stand-by capability. This means the disk drive motor is not in constant operation, but waits to be "told" when it is needed.

## ATARI 1050 DISK DRIVE

The ATARI 1050 Disk Drive unit can use a double-density recording technique. A track on a diskette created in double-density mode can store 26 sectors instead of 18.

Like the ATARI 810 Disk Drive, this drive also uses standard 5 1/4 inch flexible diskettes, but can store 130K bytes on each diskette. The ATARI 1050 Disk Drive has a built-in microprocessor to control the drive.

# DISK DRIVE OPERATION

When you insert a diskette into the disk drive, the spindle hole in the center is automatically placed on the drive hub and the diskette is seated. The circular diskette spins within its protective jacket. When you access the diskette, the magnetic head is placed over the read/write surface.

When you store data on a diskette, the disk drive converts the data it receives from the computer console into coded electrical pulses. These pulses magnetize minute areas of the oxide coating of each diskette while the diskette is spinning.

When you retrieve data from a diskette, the disk drive positions the magnetic head so the area of the diskette where the data is stored passes beneath it. The disk drive's microprocessor controls the positioning and timing of the diskette.

# ALPHABETIC DIRECTORY OF BASIC RESERVED WORDS USED WITH DISK OPERATIONS

---

**Note:** The period is mandatory after all abbreviated keywords.

| RESERVED WORD | ABBREVIATION | BRIEF SUMMARY OF BASIC STATEMENT |
|---|---|---|
| CLOSE | CL. | I/O statement used to close a disk file at the conclusion of I/O operations. |
| DOS | DO. | This command causes the menu to be displayed. The menu contains all DOS utility selections. Passes control from cartridge to DOS utilities. |
| END | | Stops program execution, closes files, and turns off sounds. Program may be restarted using CONT. (Note: END may be used more than once in a program.) |
| ENTER | E. | I/O command used to retrieve a listed program in untokenized (textual) form. If a program or lines are entered when a program is resident in RAM, ENTER merges the two programs. If you don't want programs merged, type NEW before using ENTER to load a program into RAM. |
| GET | GE. | Used with disk operation to input a single byte of data into a specified variable from a specified device. |
| INPUT | I. | This command requests data from a specified device. The default device is E: (Screen Editor). |
| LIST | L. | This command outputs the untokenized version of a program to a specified device. |
| LOAD | LO. | I/O command used to retrieve a saved program in tokenized form from a specified device. |
| NOTE | NO. | This command stores the current byte number of the file pointer in its two arithmetic variables. |
| OPEN | O. | Opens the specified file for input or output operations. Determines the type of operations allowed on a file. |
| POINT | P. | This command is used in setting the file pointer to a specified location (byte) on the diskette, relative to the beginning of the file. |

| | | |
|---|---|---|
| **PRINT** | **PR. or?** | I/O command causes output from the computer to specified output device in record format. |
| **PUT** | **PU.** | Causes output of a single byte of data; i.e., a character, from the computer to a specified device. |
| **RUN** | **R.** | Both loads and starts execution of designated filespec. |
| **SAVE** | **S.** | I/O statement used to record a tokenized version of a program in a specified file on a specified device. |
| **STATUS** | **ST.** | Calls status routine for specified device. |
| **TRAP** | **T.** | Directs execution to a specified line number in case of a program error, allowing user to maintain control of program and recover from errors. |
| **XIO** | **X.** | General I/O statement used in a program to perform DOS Menu selections and specified I/O commands. |

# NOTATIONS AND TERMINOLOGY USED WITH DOS 3

| | |
|---|---|
| **SYSTEM RESET** | Press **SYSTEM RESET** key on the keyboard. |
| **RETURN** | Press **RETURN** key on the keyboard. |
| [ ] | Brackets. Brackets enclose optional items. |
| . . . | Ellipsis. An ellipsis following an item in brackets indicates you can repeat the optional item any number of times, but are not required to do so. |
| { } | Braces. Items stacked vertically in braces indicate you have a choice as to which item you want to insert. Select only one to put in your statement or command. |
| **CAPITAL LETTERS** | Capital letters are used to indicate commands, statements, and other functions you must type exactly as they appear. |
| , . / : ; " | Punctuation marks. These punctuation marks must be typed as shown in the format of a command or statement. However, do not type brackets or braces. |
| **cmdno** | Command number. Used in XIO commands. |
| **exp** | Expression. In this manual, expressions are divided into three types: arithmetic, logical, and string expressions. |
| **aexp** | Arithmetic expression. Generally composed of a variable, function, constant, or two arithmetic expressions separated by an arithmetic operator (aop). |
| **aexp1** | Arithmetic expression 1. This arithmetic expression represents the first auxiliary I/O control byte when used in commands such as OPEN. |
| **aexp2** | Arithmetic expression 2. This arithmetic expression represents the second auxiliary I/O control byte when used in commands such as OPEN. Usually it is set to 0. If, however, you want to direct the ATARI 820 Printer to print sideways, you would set this arithmetic expression to 83. |
| **filespec** | File specification. Usually a strong expression that refers to a file and the device where it is located, e.g.,"D1:MYPROG.BAS" for a file on Drive 1. |
| **IOCB** | Input/Output Control Block (IOCB). An arithmetic expression that evaluates to a number from 1 to 7. The IOCB is used to refer to a device or file. IOCB 0 is reserved for BASIC for the Screen Editor and should only be used if the Screen Editor is not to be used. |

| | |
|---|---|
| **lineno** | Line number. A constant that identifies a particular program line in a deferred mode BASIC program. A line number can be any integer from 0 through 32767. Line numbering determines the order of program execution. |
| **var** | Variable. Any variable. In this manual, variables are classified as arithmetic variables (avar), matrix variables (mvar), or string variables (svar). |
| **avar** | Arithmetic variable. A location where a numeric value is stored. Variable names can be from 1 to 120 alphanumeric characters, but must start with an unreversed, uppercase alphabetic character. |
| **mvar** | Matrix variable. Also called a subscripted variable. An element of an array or matrix. |
| **svar** | String variable. A location where a string of characters may be stored. |

# ERROR MESSAGES AND HOW TO RECOVER

**Note:** Error messages 2 through 21 should only occur when running a BASIC program.

| ERROR No. | ERROR NAME | CAUSE AND RECOVERY |
|---|---|---|
| 2 | Insufficient Memory | You do not have enough memory to store the statement, or to dimension a new string variable. Delete any unused variable names or add more memory. (See Section 11, BASIC Reference Manual for memory conservation.) |
| 3 | Value Error | Either the expected positive integer was negative or the value was not within the expected range. |
| 4 | Too Many Variables | You have exceeded the maximum number (128) of variable names and must delete any that are no longer applicable. (See Section 11, BASIC Reference Manual.) |
| 5 | String Length Error | You have attempted to read from or write into a location past the dimensioned string size or you have used zero as a reference index. Enlarge DIM size. Do not use zero as an index. |
| 6 | Out of Data Error | You do not have enough data in your DATA statements for the READ statements. |
| 7 | Line Number Greater Than 32767 | Check line number references in statements such as GOTO and RESTORE. |
| 8 | Input Statement Error | You have attempted to input a non-numeric value into a numeric variable. Check your variable types and/or input data. |
| 9 | Array or String DIM Error | The DIM size exceeds 5460 for numeric arrays or 32767 for strings; an array or string was redimensioned; reference was made to an undimensioned array or string. |
| 11 | Floating Point Overflow/ Underflow | You have attempted to divide by zero or to refer to a number with an absolute value less than $1E-99$, or greater than or equal to $1E-98$. |
| 12 | Line Not Found | A GOSUB, GOTO, or THEN referenced a non-existent line number. |
| 13 | No Matching FOR | A NEXT statement was encountered without a matching FOR. |
| 14 | Line Too Long Error | You have exceeded the BASIC line processing buffer length. |

| ERROR NO. | ERROR NAME | CAUSE AND RECOVERY |
|---|---|---|
| 15 | GOSUB or FOR Line Deleted | A NEXT or RETURN statement was encountered and the corresponding FOR or GOSUB was deleted since the last time the program was run. |
| 16 | RETURN Error | Check your program for a missing GOSUB statement. |
| 17 | Syntax Error | The computer encountered a line with improper syntax. Fix the line. |
| 18 | VAL Function Error | The string in a VAL statment is not a numeric string. |
| 19 | LOAD Program Too Long | You don't have enough memory to load your program. |
| 20 | Device Number Error | You entered a device number that was not between 1 and 7. |
| 21 | LOAD File Error | You attempted to load a nonload file, not a BASIC tokenized file. Tokenized files are created with the SAVE command. |

**Note:** The following are input/output errors that result during the use of disk drives, printers, or other accessory devices. Further information is provided with the auxiliary hardware.

| | | |
|---|---|---|
| 128 | BREAK Abort | User hit BREAK key during I/O operation. Stops execution. |
| 129 | IOCB* Already Open | OPEN statement within a program loop or IOCB already in use for another file or device. |
| 130 | Nonexistent Device | You have tried to access a device not in the handler table; i.e., the device is undefined. This error can occur when trying to access the ATARI 850™ Interface Module without running the RS-232-C HANDLER.SYS file. Another common occurrence of this error is specifying a filename without a device; i.e., "MYFILE" instead of "D:MYFILE." |
| | | Check your I/O command for the correct device. Then load and initialize the correct handler. |

*IOCB refers to Input/Output Control Block.

| | | |
|---|---|---|
| 131 | IOCB Write-Only | You have attempted to read from a file opened for Write-Only. Open the file for read or update (read/write). |
| 132 | Illegal Handler Command | This is a CIO error code. The common code passed to the device handler is illegal. The command is either $\leq$ 2 or is a special command to a handler that hasn't implemented any special commands. Check your XIO or IOCB command code for an illegal command code. |
| 133 | Device/File Not Open | You have not opened this file or device. Check your OPEN statement or file I/O statement for a wrong file specification. |

| ERROR No. | ERROR NAME | CAUSE AND RECOVERY |
|---|---|---|
| 134 | Bad IOCB Number | You have tried to use an illegal IOCB index. For BASIC, the range is 1-7 as BASIC does not allow use of IOCB 0. The Assembler Editor cartridge requires the IOCB index to be a multiple of 16 and less than 128. |
| 135 | IOCB Write-Only Error | You have tried to write to a device or file that is open for Read-Only. Open the file for write or update (read/write). |
| 136 | End of File | Your input file is at end of file. Not more data in file. |
| 137 | Truncated Record | This error typically occurs when the record you are reading is larger than the maximum record size specified in the call to CIO. (BASIC's maximum record size is 119 bytes.) Trying to use an INPUT (record-oriented) type of command on a file that was created with PUT (byte-oriented) commands causes this problem. |
| 138 | Device Timeout | When you sent a command over the serial bus, the device did not respond within the period set by the operating system for that device command. Either the device number is wrong or the user specified the wrong device; the device is not there (wrong spec); it is unable to respond within the proper period; or it is not connected. If the device is a cassette, the tape baud rate may have been mismeasured or the tape improperly positioned. Test all connections to make sure they are secure and check the disk drive to make sure it is turned on and set for the correct drive number. Check your command for the correct drive number. Retry the command. If this error recurs, have the disk drive checked. |
| 139 | Device NAK | The device cannot respond because of bad parameters such as an unaddressable sector. The device might also have received a garbled or illegal command or received improper data from the computer. Check your I/O command for illegal parameters and retry the command. Also check your I/O cables. This is a device specific error, so refer to the documentation for that device. |
| 140 | Serial Frame Error | Bit 7 of SKSTAT in the POKEY chip is set. This means that communication from the device to the computer is garbled. |
|  |  | This is a very rare error and it is fatal. If it occurs more than once, have your device or computer checked. You can also remove the peripherals one at a time to isolate the problem. For cassettes, try the recovery suggested in Error 138. |

| ERROR NO. | ERROR NAME | CAUSE AND RECOVERY |
|---|---|---|
| 141 | Cursor Out of Range | Your cursor is out of range for the particular graphics mode you chose. Change the pixel parameters. |
| 142 | Serial Bus Overrun | Bit 5 of SKSTAT in POKEY is set. The computer did not respond fast enough to a serial bus input interrupt or POKEY received a second 8-bit word on the serial bus before the computer could process the previous word. This is a rare error. If it occurs more than once, have your computer serviced. |
| 143 | Checksum Error | The communications of the serial bus are garbled. The checksum sent by the device is not the same as that calculated for the frame received by the computer. There is no standard recovery procedure because it could be either a hardware or software problem. |
| 144 | Device Done Error | The device is unable to execute a valid command. You have either tried to write to a write-protected diskette or device, or the disk drive is unable to read/write to the requested sector. Remove the write-protected tab or turn off the write-protect switch. See specific manuals for other device. |
| 145 | Illegal Screen Mode | You have tried to open the Screen Editor with an illegal graphics mode number. Check your graphics mode call or the aux2 byte in the IOCB. |
| 146 | Function Not Implemented | The handler does not contain the function; e.g., trying to PUT to the keyboard or issuing special commands to the keyboard. Check your I/O command for the right command and the correct device. |
| 147 | Insufficient RAM | Not enough RAM for the graphics mode you selected. Add more memory or use a graphics mode that doesn't require as much memory. |
| 160 | Drive Number Error | You specified a drive number that was not 1-8, did not allocate a buffer for the drive, or your drive was not powered up at boot time. Refer to Section 1 of this manual. Check your filespec or byte 1802 for number of drive buffers allocated. |
| 161 | Too Many OPEN Files | You don't have any free system buffers to use on another file. Make sure no files are open that should not be open. |
| 162 | Disk Full | You don't have any more free blocks on this diskette. Use a different diskette that has free blocks. |
| 163 | Unrecoverable System I/O Error | This error means that the File Manager has a bug in it. Your DOS or the diskette may be bad. Try using other DOS. |
| 165 | File Name Error | Your file specification has illegal characters in it. Check filespec and remove illegal characters. |

| ERROR NUMBER | ERROR NAME | CAUSE AND RECOVERY |
|---|---|---|
| 167 | File Protected | You have tried to access a protected file for purposes other than reading it. Use DOS Menu option U to unlock the file and retry your command. |
| 168 | Device Command Invalid | You issued an illegal command to the device software interface. Check the documentation for that device and retry the command. |
| 169 | Directory Full | You have used all the space allocated for the Directory. |
| 170 | File Not Found | You have tried to access a file that doesn't exist in the diskette's Directory. Use DOS Menu option F to check the correct spelling of the filename and to be sure it is on the diskette you are accessing. |
| 171 | POINT Invalid | The byte count in the POINT call was greater than the number of bytes in the file. Check the parameters in your POINT statement. |
| 173 | Bad Sectors at Format Time | The disk drive has found bad sectors while formatting a diskette. Use another diskette, as you cannot format a diskette with bad sectors. If this error occurs with more than one diskette, your disk drive may need repair. |
| 174 | Duplicate Filename | You performed a RENAME funct that would result in two files of the same name on the diskette. Since DOS 3 is not able to tell which of them you want to use for the next operation involving that name, it refuses to execute this specific rename command. Solution: Use a different name for the rename, or if you no longer need the other file, erase it. |
| 175 | Bad Load File | The file which you asked DOS 3 to load is not a load-type file. |
| 176 | Incomaptible Format | You are trying to perform a DOS 3 operation using a DOS 2 diskette. Use the function called ACCESS DOS 2 to translate your file for further use with DOS 3. |
| 177 | Disk Structure Damaged | DOS 3 does not recognize the files on the disk due to some form of damage to the disk, whether electrical, mechanical or use of a non-DOS 3 disk itself. |
| | | **Note:** the errors 2 through 10 listed below can occur only during DOS 3 menu selection operations. |
| 2 | No Command | No file with an extender of .CMD exists in drive 1. |
| 3 | Input Required | You have tried to enter a blank for the new name in a Rename command or some other similar item where there is no default. If an input is required, you must provide it. Check the HELP screens to see what kind of input is needed. |

| ERROR No. | ERROR NAME | CAUSE AND RECOVERY |
|---|---|---|
| 4 | No Cartridge | You have executed the TO-CARTRIDGE function with no cartridge present. You have to power-down and install the cartridge to be able to execute this function. |
| 5 | I/O Error | Some I/O error (e.g., printer is not on line for a print operation) occurred. |
| 6 | Invalid End Address | You have given an END address for the SAVE function which is less than the start address. Check you entry and try again. |
| 7 | Error Loading MEM.SAV | The system has been unable to restore the memory using the MEM.SAV file. The program which you had in memory when you typed DOS, is lost. This error may mean a faulty diskette or a dirty drive. Check that the file MEM.SAV is on the diskette in drive 1. |
| 8 | Error Saving MEM.SAV | Something has happened while the system was trying to write the MEM.SAV file. This file is not valid on the disk. You might try installing a different disk for the next time. There is no possible recovery from this problem. |
| 9 | Drive Input Error | You supplied an invalid device specification. |
| 10 | Filename Input Error | You supplied an invalid filename. |
| 128 | BREAK abort | You have touched the BREAK key during one of the operations and the function was stopped. Decide which function you wish and repeat it, without a BREAK. |

# DOS 3 MEMORY MAP

| ADDRESS | | CONTENTS | |
|---------|---|----------|---|
| **Decimal** | **Hexadecimal** | | |
| 65535 | FFFF | | |
| | | OPERATING SYSTEM | |
| 49152 | C000 | | |
| 49151 | BFFF | | |
| | | CARTRIDGES | |
| 32768 | 8000 | | |
| 32767 | 7FFF | | |
| | | SCREEN DISPLAY AREA | |
| varies | | | |
| varies | | | HIMEM** |
| | | USER PROGRAM AREA | |
| varies | | | LOMEM* |
| | | System Buffer N | UTILITY PROGRAMS |
| | | . | |
| | | . | and |
| | | . | BUFFER AREA |
| | | System Buffer 2 | RESERVED |
| 6780 | 1A7C | System Buffer 1 | FOR DOS 3 |
| 6779 | 1A7B | | |
| | | KEYBOARD COMMAND PROCESSOR (KCP) (RAM RESIDENT PORTION OF KCP) | |
| 5728 | 1660 | | |
| 5727 | 165F | | |
| | | FILE MANAGEMENT SYSTEM (FMS) | |
| 1792 | 0700 | | |
| 1791 | 06FF | | |
| | 0600 | USER RAM | |
| | 05FF | | |
| 0 | 0000 | OPERATING SYSTEM RAM | |

\* Varies with the number of System Buffers reserved. (See INIT Function.)
\*\*Depends on which Graphics Mode is currently in use.

**Note 1:** For given System Buffer allocation, LOMEM can be determined by PEEK Locations 2E7 (LOW) and 2E8 (HIGH) Hex or 743 (LOW) and 744 (HIGH) Decimal.

**Note 2:** To determine the amount of User Program Area available or HIMEM, you can either make use of the BASIC FRE(O) instruction or PEEK Locations 2E5 (LOW) and 2E6 (HIGH) Hex or 741 (LOW) and 742 (HIGH) Decimal.

# HEXADECIMAL TO DECIMAL CONVERSION TABLE

APPENDIX E

| | FOUR HEX DIGITS | | | | | | |
|---|---|---|---|---|---|---|---|
| | | 4 3 2 1 | | | | | |
| **HEX** | **DEC** | **HEX** | **DEX** | **HEX** | **DEC** | **HEX** | **DEC** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 4096 | 1 | 256 | 1 | 16 | 1 | 1 |
| 2 | 8192 | 2 | 512 | 2 | 32 | 2 | 2 |
| 3 | 12288 | 3 | 768 | 3 | 48 | 3 | 3 |
| 4 | 16384 | 4 | 1024 | 4 | 64 | 4 | 4 |
| 5 | 20480 | 5 | 1280 | 5 | 80 | 5 | 5 |
| 6 | 24576 | 6 | 1536 | 6 | 96 | 6 | 6 |
| 7 | 28672 | 7 | 1792 | 7 | 112 | 7 | 7 |
| 8 | 32768 | 8 | 2048 | 8 | 128 | 8 | 8 |
| 9 | 36864 | 9 | 2304 | 9 | 144 | 9 | 9 |
| A | 40960 | A | 2560 | A | 160 | A | 10 |
| B | 45056 | B | 2816 | B | 176 | B | 11 |
| C | 49152 | C | 3072 | C | 192 | C | 12 |
| D | 53248 | D | 3328 | D | 206 | D | 13 |
| E | 57344 | E | 3584 | E | 224 | E | 14 |
| F | 61440 | F | 3840 | F | 240 | F | 15 |

*Use this table to convert up to four hex digits.

For example, to convert the hex number 1234 to decimal, add the entries from each of the four columns in the table. For 1, use the column number 4, and so on.

```
1234 hex. = 4096
          +  512
          +   48
          +    4

          4660 dec.
```

Other examples:

```
EEDD hex. = 57344
          +  3584
          +   208
          +    13

          61149 dec.

  AB hex. = 160
          +  11

          171 dec.
```

# HOW TO SPEED UP DATA TRANSFERS TO DISK DRIVE

Your new DOS 3 has the ability to Write and Read Verify (a safety technique that should be used whenever improved reliability is more important than rapid data transmissions). This is the way your DOS 3 Master Diskette is shipped to you. To save time, however, the information can be written to the diskette without a Read Verify. During disk initialization, the File Management Subsystem may be modified to either use write with verify or write without verify. Write without Read Verify is faster, but may not be as reliable.

To alter the version of DOS stored on diskette so that your custom version will always boot in, simply type DOS and then use the I (Init Disk) command from the Disk Operating System Menu and answer Yes to the question "Modify FMS parameters?" to store the new version of DOS from RAM onto your diskette.

# MAJOR DIFFERENCES BETWEEN DOS 3 AND DOS 2

---

1. The DOS 3 file manager and buffers now take up less space than the DOS 2 equivalents. All utilities, such as COPY, INIT, DUPLICATE (all UTL files) are called into memory only as needed. Each is maintained in a separate file.
2. DOS 3 provides a direct method for the user to modify the FMS parameters (shown in the description of the INIT function).
3. DOS 3 provides an online HELP feature that was not available on DOS 2.
4. The NOTE and POINT commands return a pointer number relative to the start of a file (byte 0) rather than an absolute sector and byte location within the sector. If both the sector variable and the byte variable are stored for future use, program compatibility can be maintained. In other words, if the DOS 2 conventions are followed, programs can run on either DOS 2 or DOS 3.

# STRUCTURE OF A COMPOUND FILE

## COMPOUND FILE STRUCTURE USING COPY/APPEND

| | Decimal | Hex | |
|---|---|---|---|
| Byte No. | No. | No. | Description |
| 1 | 255 | FF | Identification Code (PART 1) |
| 2 | 255 | FF | |
| 3 | 0 | 00 | Starting Address (PART 1) |
| 4 | 80 | 50 | |
| 5 | 31 | 1F | Ending Address (PART 1) |
| 6 | 80 | 50 | |
| . | . | . | DATA (PART 1) |
| . | . | . | 32 Bytes |
| . | . | . | |
| 38 | 255 | FF | Identification Code (PART 2) |
| 39 | 255 | FF | |
| 40 | 32 | 20 | Starting Address (PART 2) |
| 41 | 80 | 50 | |
| 42 | 143 | 8F | Ending Address (PART 2) |
| 43 | 80 | 50 | |
| . | . | . | DATA (PART 2) |
| . | . | . | 112 Bytes |
| . | . | . | |

## COMPOUND FILE STRUCTURE USING SAVE

| | Decimal | Hex | |
|---|---|---|---|
| Byte No. | No. | No. | Description |
| 1 | 255 | FF | Identifier Code |
| 2 | 255 | FF | |
| 3 | 00 | 00 | Starting address (PART 1) |
| 4 | 80 | 50 | |
| 5 | 31 | 1F | Ending address (PART 1) |
| 6 | 80 | 50 | |
| . | . | . | Data (PART 1) |
| . | . | . | 32 Bytes |
| . | . | . | |
| 38 | 32 | 20 | Starting address (PART 2) |
| 39 | 80 | 50 | |
| 40 | 143 | 8F | Ending address (PART 2) |
| 41 | 80 | 50 | |
| . | . | . | Data (PART 2) |
| . | . | . | 112 Bytes |
| . | . | . | |

# GLOSSARY
# OF TERMS

| | |
|---|---|
| **Access** | The method (or order) in which information is read from, or written to diskette. See Sequential Access and Random Access. |
| **Address** | A location in memory, usually specified by a two-byte number in hexadecimal or decimal format. (Maximum range is 0-FFFF hexadecimal.) |
| **Alphanumeric** | The letters A-Z and the numbers 0-9, and/or combinations of letters and numbers. Specifically excludes graphics symbols, punctuation marks, and other special characters. |
| **Array** | A one-dimensional set of elements that can be referenced one at a time or as a commmplete list by using the array variable name and one subscript. Thus the array B, element number 10 would be referred to as B(10). Note that string arrays are not supported by BASIC, but you can pick up each element within a string; for example, A$(10). All arrays must be dimensioned before use. A matrix is a two-dimensional array. |
| **ATASCII** | The method of coding used to store external data. In ATASCII (which is a modified version of ASCII, the American Standard Code for Information Interchange), each character and graphics symbol, as well as most of the control keys, has a number assigned to represent it. The number is a one-byte code (decimal 0-255). See the ATARI BASIC Reference Manual for table. |
| **AUTORUN.SYS** | Filename reserved for use by user. If AUTORUN.SYS is on disk being booted, it will be loaded into RAM. |
| **Baud Rate** | Signaling speed or speed of information interchange in bits per second. |
| **Binary** | A number base system using the digits 0 and 1. |
| **Bit** | Abbreviation of "binary digit." The smallest unit of information, represented by the value 0 or 1 in binary. |
| **Block** | A total of 8 sectors, treated as a group by the File Manager. |
| **Boot** | This is the initialization program that "sets up" the computer when it is powered up. At conclusion of the boot (or after "booting up"), the computer is capable of loading and executing high level programs. |

| | |
|---|---|
| **Break** | To interrupt execution of a program. Pressing the BREAK key causes a break in execution. |
| **Buffer** | A temporary storage area in RAM used to hold data for further processing, input/output operations, and the like. |
| **Byte** | Eight bits. A byte can represent one character. A byte has a range of 0-255 (decimal) or 0-FF (hex). |
| **Cartridge** | An assembly containing a read-only memory (ROM). Usually intended to contain an executable program. |
| **CIO** | Central Input/Output Subsystem. The part of the OS that handles input/output. |
| **CLOSE** | To terminate access to a disk file. Before further access to the file, it must be opened again. See OPEN. |
| **CONVERT.UTL** | File containing the ACCESS DOS 2 utility. |
| **COPY.UTL** | File containing the COPY utility. |
| **Data** | Information of any kind, usually a sequence of bytes. |
| **Debug** | To isolate and eliminate errors from a program. |
| **Decimal** | A number base system using the digits 0 through 9. Decimal numbers are stored in Binary Coded Decimal format (or in binary format) in the computer. See Bit, Binary, Hexadecimal, and Octal. |
| **Default** | A condition or value that exists or is caused to exist by the computer until it is told to do something else. For example, the computer defaults to GRAPHICS 0 until another graphics mode is entered. |
| **Delimiter** | A character that marks the start or finish of a data item but is not part of the data. For example, the quotation marks ('') are used by most BASIC systems to delimit strings. |
| **Density** | The closeness of space distribution on a storage medium; the number of sectors per diskette. Single-density is 720 sectors per diskette; double-density is 1040 sectors per diskette. |
| **Destination** | The device or address that receives data during an exchange of information (and especially an I/O exchange). See Source. |
| **Directory** | A portion of a diskette containing a summary of files contained on a diskette by filename, file size, and location. |
| **Disk Drive** | A device for reading and writing disks. |
| **Diskette** | A small disk. A record/playback medium like tape, but made in the shape of a flat disk and placed in an envelope for protection. The access time for a diskette is much faster than for tape. |
| **DOS** | Abbreviation for Disk Operating System. The software or programs that facilitate use of a disk drive system. |

| | |
|---|---|
| **Drive Specification or Drivespec** | Part of the filespec that tells the computer which disk drive to access. If this is omitted the computer will assume Drive 1. |
| **Drive Number** | An integer from 1 to 8 that specifies the disk drive to be used. |
| **DUPDISK.UTL** | File containing the DUPLICATE utility. |
| **End of File** | A marker that tells the computer that the end of a certain file has been reached. |
| **Entry Point** | The address where execution of a machine-language program or routine is to begin. Also called the run address. |
| **File** | An organized collection of related data. A file is the largest grouping of information that can be addressed with a single name. For example, a BASIC program is stored on diskette as a particular file, and may be addressed by the statements SAVE or LOAD (among others). |
| **File index** | Listing of file name and size directory information. |
| **Filename** | The alphanumeric characters used to identify a file. A total of eight numbers and/or letters may be used. |
| **Filename Extender or Extension** | From 0 to 3 additional characters used following a period (required if the extender is used) after the filename. For example, in the filename PHONLIST.BAS the letters "BAS" comprise the extender. |
| **File Pointer** | A pointer to a location in a file. Each file has its own pointer. |
| **Filespec** | Abbreviation for file specification. A sequence of characters which specifies a particular device and filename. Consists of a drive spec and a file name and optional file name extension. |
| **FMS.SYS** | Filename containing the File Management System. |
| **Format** | To organize a new or magnetically (bulk) erased diskette onto tracks and sectors. When formatted, each diskette contains 40 circular tracks, with 18 or 26 sectors per track. Each sector can store up to 128 bytes of data. |
| **HANDLER.SYS** | File name reserved for use by DOS 3 similar to AUTORUN.SYS. |
| **HELP.TXT** | File containing help information. |
| **HELP.UTL** | File containing the HELP utility. |
| **Hexadecimal or Hex** | Number base system using 16 alphanumeric characters 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E, and F. Addresses and byte values are sometimes given in hex. |
| **Indexed Addressing** | See Random Accessing. |
| **INIT.UTL** | File containing the INIT utility. |

| | |
|---|---|
| **Input** | To transfer data from outside the computer (say, from a diskette file) into RAM. Output is the opposite, and the two words are often used together to describe data transfer operations: Input/Output or just I/O. Note that the reference point is always the computer. Output always means away from the computer, while Input means into the computer. |
| **INPUT** | A BASIC command used to request either numeric or string data from a specified device. |
| **I/O** | Input/Output. |
| **IOCB** | Input/Output Control Block. A section of RAM reserved for addressing an input or output device and processing data received from it or for addressing and transferring data to an output device. |
| **iocb** | An arithmetic expression that evaluates to a number between 1 and 7. This number is used to refer to a device or file. |
| **KCP.SYS** | File containing the resident portion of the keyboard command processor. |
| **KCPOVER.SYS** | File containing non-resident portion of the keyboard command processor. |
| **Kilobyte or K** | 1024 bytes. Thus a 16K RAM capacity actually gives us 16 times 1024 or 16,384 bytes. |
| **Least Significant Byte** | The byte in the rightmost position in a number of a word. |
| **LOAD** | To read from a diskette file into memory. Usually consists of an executable object file or direct machine-readable code. |
| **Machine Language** | The instruction set for the particular microprocessor chip used, which for ATARI is the 6502. |
| **Most Significant Byte** | The byte in the leftmost position in a number or a word. |
| **Null String** | A string consisting of no character whatever. For example, A$""""stores the null string in variable A$. |
| **Object Code** | Machine language derived from ''source code,'' typically from assembly language. |
| **Octal** | The octal numbering system uses the digits 0 through 7. |
| **OPEN** | To prepare a file for access by specifying whether an input or output operation is to be conducted, and specifying the filespec. |
| **Parameter** | Variables in a command or function. |
| **Peripheral** | An I/O device. |
| **POKEY** | A custom I/O chip that manages communication on the serial bus. |

| | |
|---|---|
| **Random Access** | The method of reading data from a diskette directly from the byte and sector where it was stored without having to read the entire file sequentially. |
| **Record** | A block of data, delimited by an EOL (End-Of-Line, 9B Hex) character. |
| **SAVE** | To write from memory to diskette. Consists of either machine-readable code or an executable program. |
| **Sector** | A sector is the smallest block of data that can be written to a disk file or read from a file. Each sector can store 128 bytes of data. |
| **Sequential Access** | The method of reading each byte from the diskette in order, starting from the first byte in the sector. |
| **Source** | The device or address that contains the data to be sent to a destination. See Destination. |
| **Source Code** | A series of instructions, written by the user. Needs to be translated to machine language in order to be executed. |
| **String** | A sequence of letters, characters, stored in a string variable. The string variable's name must end with a $. |
| **System buffer** | A buffer used by FMS. Each device and each open file require one system buffer. |
| **System Diskette** | An exact backup copy of original Master Diskette. Always use backup copies of your Master Diskette instead of the original. Keep backup copies of all important data and program diskettes. |
| **Tokenizing** | The process of interpreting textual BASIC source code and converting it to the internal format used by the BASIC interpreter. |
| **Track** | A circle on a diskette used for magnetic storage of data. Each track has 18 or 26 sectors, each with 128-byte storage capability. There are a total of 40 tracks on each diskette. |
| **Variable** | A variable may be thought of as a box in which a value may be stored. Such values are typically numbers and strings. |
| **Write-Protect** | A method of preventing the disk drive from writing on a diskette. ATARI diskettes are write-protected by covering a notch on the diskette cover with a small sticker. |

| ERROR CODE NO | ERROR CODE MESSAGE |
|---|---|
| 2 | No command File Found |
| 3 | Input Required |
| 4 | No Cartridge |
| 5 | I/O Error |
| 6 | Invalid End Address |
| 7 | Error Loading MEM.SAV |
| 8 | Error Saving MEM.SAV |
| 9 | Device Specification Input Error |
| 10 | Filename Input Error |
| 128 | Break Abort |
| 130 | Nonexistent Device |
| 136 | End of File |
| 138 | Device Time Out |
| 139 | Device NAK |
| 140 | Serial Bus Input Framing Error |
| 141 | Cursor Out of Range |
| 142 | Serial Bus Data Frame Overrun Error |
| 143 | Serial Bus Data Frame Checksum Error |
| 144 | Device Done Error |
| 160 | Device Number Error |
| 161 | Too Many Open Files |
| 162 | Disk Full |
| 163 | System I/O Error |
| 165 | Filename Error |
| 167 | File Protected |
| 168 | Special Command Invalid |
| 169 | Directory Full |
| 170 | File Not Found |
| 171 | Point Invalid |
| 173 | Bad Sectors at Format Time |
| 174 | Duplicate File |
| 175 | Bad Load File |
| 176 | Format Incompatible |
| 177 | Disk Structure Error |

# INDEX

**Boldface** page numbers refer to sections discussing the item indexed.

# DISK OPERATING SYSTEM III
# REFERENCE MANUAL

# ATARI DOS 3 REFERENCE MANUAL ERRATA
## 05/01/84

MANUAL CORRECTIONS

PAGE 14, In the 'INTEREST' program, line 200 reads:

    200 LET A=P*(1+R/100) N

Change to:

    200 LET A=INT(P*(1+R/100) $\wedge$ N+0.5)

PAGE 58, 'code 13', reads:

    13 = input and Output operation; positions file pointer to EOF; writing past EOF is allowed.

This line should be:

    13 = input and Output operation; positions file pointer to BEGINNING of file; writing past EOF is allowed.

PAGE 60, gives information on Random Access methods. The program indicates that only the 'X' parameter is significant, actually both the 'X' and 'Y' parameters are significant. For a correct demonstration of the Random Access NOTE and POINT commands contact ATARI Customer Relations at the address listed below.

PAGES 65 and 66, the 'BIRTHDAY' program requires several changes:

Line 140 reads:

    140 IF MONTH 1 OR MONTH 12 THEN?ERR$:GOTO 100

Change to:

    140 IF MONTH $<$ 1 OR MONTH $>$ 12 THEN?ERR$:GOTO 100

Line 240 reads:

    240 NEXT

Change to:

    240 NEXT I

Line 310 reads:

    310 IF A$ $><$ "YES" THEN GOTO 20

Change to:

    310 IF A$ $<>$ "YES" THEN GOTO 20

Line 340 reads:

    340 ? "PLEASE TYPE PERSONS BIRTHDAY (MM-DO-YY)"

Change to:

    340 ? "PLEASE TYPE PERSONS BIRTHDAY (MM-DD-YY)"

Line 390 reads:

    390 TRAP 400:XIO 36,# 3,0,0, FILE$:OPEN #,9,0,FILE$:GOTO 410

Change to:

    390 TRAP 400:XIO 36,#2,0,0, FILE$:OPEN #2,9,0,FILE$:GOTO 410

Line 620 reads:

    620 IF MONTH 12 THEN MONTH=1

Change to:

    620 IF MONTH $>$12 THEN MONTH=1

Line 700 reads:
          700 EOF=0;GOTO 600
Change to:
          700 EOF=0:GOTO 600
Line 3000 should be omitted.

## ATARI DOS 3 ANOMALIES

Early versions of ATARI DOS 3 used a random access method that was incompatible with large files. To determine if you have an early version, boot your copy of ATARI DOS 3 with ATARI BASIC, and execute the following command from ATARI BASIC:

## PRINT PEEK (1816)

If the value returned is '53', your copy of ATARI DOS 3 is a current version and does not require modification.

If the value returned is '51' or '56', please contact ATARI Customer Relations at one of the toll free numbers or address listed below, to obtain a program which will update ATARI DOS 3 to the current revision level.

If you want to receive additional information on the structure of an ATARI DOS 3 diskette, and how ATARI DOS 3 manages diskette files, or if you have any other questions concerning ATARI DOS 3, please contact ATARI Customer Relations at one of the following toll free numbers, or address:

> ATARI Customer Relations
> PO BOX 61657
> 1312 Crossman Ave.
> Sunnyvale, CA 94088
>
> (800) 538-8543 Outside California
> (800) 672-1404 Inside California