INDUS GT™

ATARI

DOS XL™
Operator's Guide
and
Reference Manual

DOS XL[tm]

Operator's Guide and Reference Manual

for Drivers of the Indus GT[tm]

Atari[tm] Compatible Diskette Drive

Adaptation by Keith S. Burgoyne

This manual was last revised on February 1, 1985

TRADEMARK NOTICES

The following trademarked names are used in various places within this manual, and credit is hereby given:

INDUS GT is a trademark of Indus Systems of Chatsworth, California.

DOS XL, OS/A+, BASIC XL, ACTION!, BASIC A+, MAC/65, and C/65 are trademarks of Optimized Systems Software Incorporated of San Jose, California.

Atari, Atari Home Computers, Atari Writer, Atari 810 Disk Drive, Atari 1050 Disk Drive, and Atari 850 Interface Module are trademarks of Atari Corporation of Sunnyvale, CA.

PREFACE

DOS XL and Synchromesh is the result of the efforts of several persons, and we believe that proper credit should be given. The original version of the DOS XL Console Processor and the original version ("version 2") of the DOS XL File Manager System (which is, of course, identical with Atari's DOS 2.0S) were written by Paul Laughton. Later version of all other portions of DOS XL are primarily the work of Mark Rose, of Optimized Systems Software, with the collaboration of Bill Wilkinson and Mike Peters.

The Synchromesh support enhancements to DOS XL were primarily labored over by Greg Ferris with assistance from Mark Rose, Bill Wilkinson, and Mike Peters, all of Optimized Systems Software under contract from Indus Systems.

The GTSYNC.COM Synchromesh utility, which includes the Synchromesh extensions to the Indus GT drive's control ROMs, as well as the Indus GT's original control ROM code were developed by Keith Burgoyne of Indus Systems.

DOS XL SERVICE AND SUPPORT POLICIES

Indus Systems has worked to bring you products which will give you years of reliable service and enjoyment. As with any software product, though, errors or omissions can and do occur. You may rest assured that, if you have a problem, every reasonable effort shall be made to help you.

In order to receive full support, you will need to direct any questions and/or problem reports about DOS XL strictly to Indus Systems. In many cases, Indus Systems has made special enhancements to the copies of DOS XL distributed with the Indus GT diskette drives in order to increase the operating system's already extensive capabilities. Because of these enhancements, Indus GT users should always contact Indus Systems for assistance and never Optimized Systems Software.

Since DOS XL is only distributed by Indus Systems to Indus GT users as a sub-licensed product, you must sign and return the Indus System's Sub-License Agreement included with your DOS XL before we can respond to your inquiries.

If you have a quick question or simply a procedural problem, you may call the Consumer Support staff at Indus Systems. Full consumer assistance is provided between the hours of 9:00 AM to 4:00 PM Pacific Time.

Although Indus Systems attempts to provide the best consumer support possible, please understand that many consumer problems can tend to be of a nature which the support staff may have never before encountered and may require a little research by our staff. Because of this, the person providing you with assistance may not be able to answer your questions in a 5-minute telephone call. So, if your problem is such that it is not easy to describe, you are invited to mail us a letter and include a diskette or computer printout detailing or demonstrating your difficulty. Always remember to include your return address, telephone number(s), the hours (for your time zone) you can be reached at the various telephone number(s) provided, and also the name and city of the Indus GT dealer from whom you purchased your Indus GT(s). In many cases, our support staff may wish to contact your dealer should they determine that local assistance would be of the best benefit.

ABOUT DOS XL AND THIS EDITION OF THE DOS XL MANUAL

DOS XL is the latest in a series of Disk Operating Systems produced by Optimized Systems Software, Inc.

DOS XL version 2.3 is a direct successor to and completely file compatible with:

Atari DOS 2.0S
OS/A+ Version 2.0
OS/A+ Version 2.1
DOS XL Version 2.2

This edition of the DOS XL manual has been issued as what we hope is both a user-friendly "manual" to the more commonly used features of the operating system as well as a true "reference manual" for the entire DOS XL Disk Operating System.

What parts of the DOS XL reference manual you read first should depend on your experience level and your purposes:

If you will never program in any language, you may not need to read any more than the "Introduction" through "The DOS XL Menu".

If you are an Atari BASIC programmer, you should definitely read "Using DOS XL with the Atari BASIC Cartridge" as well as Atari's BASIC reference manual before you start using DOS XL with Atari BASIC.

If you are an assembly language programmer, we would suggest reading this entire manual, paying special attention to "Using Assembly Language with DOS XL".

Finally, if you would like to automate DOS XL, allowing it to do several tasks for you while your computer is unattended, you need to read "Controlling DOS XL Without Being There".

Of course, regardless of your experience level or purposes, if you get tired of the restrictions of the DOS XL menu, you should read "Using the DOS XL Command Processor".

Whatever you choose to do, we hope that this manual and reference manual will help you. Written suggestions about these manuals are always helpful and carry much more impact than verbal comments. Your letters are always welcome.

Section 0: YOUR INDUS GT AND DOS XL

0.1 How Do You Get Started Using Your Indus GT and DOS XL?

It seems that new Indus GT users, whether or not the Indus GT is their first Atari diskette drive, always want to do one thing right away: see their new Indus GT operate in double density. Although we at Indus Systems would really recommend that every new user start with the DOS XL basics in order to get familiar with DOS XL and their Indus GT, we hereby submit to the majority's desires and provide, right here in the front of this manual, some quick instructions on how to get your Indus GT to operate in double density.

The first step in using double density is to create a double density version of your single density DOS XL System Master Diskette which came with your Indus GT. Which sections of this manual you need to read in order to do this depends upon your own experience level.

If you have never used any diskette drive or DOS before, you should read all of the "Introduction" section. This section will provide you with some background information which the rest of this manual will expect you to know. In addition, you should read the "Getting Started With DOS XL" section to find out how to get DOS XL started on your Atari.

If you are already familiar with using a non-Atari DOS but not a DOS on the Atari, you should read "What About Double Density?". Even if you are already experienced with single and double density on non-Atari DOS's, be sure to read the "What About Double Density?" section. Atari DOS's, in general, do not work with double density like most computers' DOS's.

All users should read "How Do You Keep Your DOS XL Diskette Safe?". Although our consumer support staff is always very nice and polite to any user who calls, you might find yourself a little embarassed calling us because you've accidentally ruined your DOS XL System Master Diskette.

All users should also read "How Do You Make a Double Density DOS XL Diskette?". This section explains how to create a diskette which is in double density; but don't stop there. Once you have a double density DOS XL system master diskette, you can simply boot this diskette and your drive will be operating in double density.

Section 0: YOUR INDUS GT AND DOS XL

Once you've followed all the directions in the sections of this manual indicated above, you can feel very confident that your Indus GT is operating in its normal finely-tuned state.

PROBLEM PREVENTOR: Never use the front panel buttons on your Indus GT to select the drive's emulation mode (as described in the Indus GT drive manual) when using DOS XL. There is absolutely no case in which this would be proper. DOS XL provides its own commands for changing your Indus GT's emulation mode (or operating density), and these commands also notify DOS itself of a density change which the Indus GT's front panel switches do not. Also, care must be used when changing diskettes in your Indus GT if the change involves two diskettes of different densities. Such a change is permitted, but a density change command must be given to DOS XL to notify it of the change.

PROBLEM PREVENTOR: Note that most of the instructions on how to convert to using double density with your Indus GT fall under the major section "Some Advanced Capabilities of DOS XL". If you are a new user, please take note of this. Once you've had fun running your Indus GT in double density, please go back to the more simpler sections of this manual to learn the basics of operating DOS XL before leaping too far ahead. Please learn to walk before you try to run.

0.2 What About the Atari 1050's Double Density?

Your Indus GT does support the Atari 1050 double density diskette format, but DOS XL does not. What does this mean? It means that if you have a copy of Atari's DOS 3.0, you can use it with your Indus GT in both single and 1050 double density modes. DOS XL does not support the 1050's double density format because this format provides only about 1.4 times (40% more) the storage capability of standard single density. DOS XL's double density format provides a full 2.0 times (100% more) the storage over single density. Whenever this manual refers to double density, it is referring to the 2.0 times double density and never the 1.4 times double density.

TABLE OF CONTENTS

Section 1: INTRODUCTION

This manual is intended for both the beginning and advanced user of the DOS XL operating system. Many of the advanced features of DOS XL which are not required for normal usage are discussed in later sections of this manual so as to avoid confusing first-time users. In many cases, early sections of this manual will discuss the "simpler side" of some of the more advanced features and will then direct you to later discussions for the "advanced side" of that particular feature.

1.1 What is Required to Use DOS XL?

DOS XL will work equally well on all models of Atari computers, provided the model you are using has at least 48K of memory inside.

1.2 Why Do You Need DOS XL?

The purpose of DOS XL is to provide a way for your Atari computer (and you) to communicate with your disk drives, printer, and other peripherals. DOS XL contains commands and utilities which allow you to:

1. Organize the information and programs with which you are working into "files". This organizing is similar to the way you would organize information printed on paper using file folders and a filing cabinet.

2. Provide an easy means by which you can access this information whenever needed.

3. Make use of various pre-written specialized application programs (like word processors, spread sheets, and data managers) and programming tools (like higher powered BASICs, machine/assembly language processors, and program "debugging" aids).

4. Go from using DOS XL itself, to using a cartridge inserted into your

Atari, to using any of the thousands of programs sold on diskette, back to using DOS XL itself.

But, in order for DOS XL to know exactly what you want it to do at any given moment, you need to "command" it. And "command" is truly the correct word for how you talk (using your Atari's keyboard) to DOS XL. Whatever you command DOS XL to do, it will do provided it knows how to do it and provided you have commanded it in a way it understands. Remember: DOS XL never actually "thinks" for itself (although beginners often find this hard to believe), it only does what you tell it to do regardless of whether or not what you told it is actually what you wanted it to do. Explaining to you exactly how to command DOS XL to get it to do your bidding is this manual's sole purpose in life.

1.3 So What is a (Diskette) "File" Anyway?

Much like the way a phonograph record or a cassette tape can hold a number of songs, a single diskette can hold many distinct files of information (up to 64 files per diskette). These files can hold programs or data in text (human readable), binary (computer readable), or several other forms. Just like songs on a phonograph record, diskette files must have names so that you can tell DOS XL exactly which file you wish to use. This is slightly different from that slow and old fashioned method of saving files on cassette using one of the Atari Program Recorders, since cassette files did not have to be given a name. Often, DOS XL and also this manual will refer to a "file name" as "file specification", or simply "filespec". They all essentially mean the same thing. "File specification" simply refers to your "specifying a file's name".

The rules for creating a file name which DOS XL will understand (a "valid" file name) are as follows:

> The file name is divided into two parts: the first part is referred to as the "primary name" and the second part is referred to as an "extension". When you create a new file name, you should make the "primary name" section something which will remind you exactly what information you have placed into that file. For instance, "CHECKS" might refer to a file containing information relating to balancing your checkbook. The extension is normally used to indicate the type of format in which the information in that file is saved. For instance, "TXT" would refer to a

text file containing a letter you have written using a word processor. Many specialized applications programs will select special "standard" extensions for you.

The primary name section of the file name can consist of up to eight capital (upper case) letters (A-Z, lower case not permitted) or numbers (0-9) arranged in just about any order your little heart desires. There is, however, one restriction: the first letter of the primary name must be a letter and not a number. The primary name section can be as short as one letter, but at least one letter must be used.

The extension section of the file name can consist of up to three letters or numbers arranged in any order, and can even start with a number. The extension is completely optional; DOS XL does not require you to specify one if you don't want one. If you do use an extension following the name section, always separate the two sections with a period (.). For instance, "CHECKS.TXT".

The following are valid file names: GEORGE, TEMP.ABS, PROG1.SAV, SORT123, COPY.COM. The following file names would not be accepted by DOS XL (illegal file names): NAMETOOLONG, TEMP.LONGEXTENSION, 1NUMBER.1ST, lowercas, NO-DASH.

The computer "hackers" who have been beating on computer keyboards for years have a sort of "standard" set of extensions which most people eventually learn sometime during their years of computer usage. To save you from waiting these years, here's a list of some of the more common extensions:

BAS for a "SAVE"d BASIC program (MYPROG.BAS)
LIS for a "LIST"ed BASIC program (MYPROG.LIS)
COM for a DOS XL command/utility program (COPY.COM)
EXC for a DOS XL execute file (STARTUP.EXC)
SYS for a DOS XL system file (DOS.SYS)
TXT for a wordprocessor or editor text file (LETTER.TXT)

In some cases, file names must be preceded by a drive specifier which tells DOS XL on which drive to search for a particular file. The format of a device specifier is:

Dn:

(or)

D:

where n is a digit from 1 to 4, depending on how many drives you have. If you just specify D:, drive 1 is assumed (this is very useful if you only have one drive). For example, D1:MYPROG.BAS tells DOS XL to use the file MYPROG.BAS on drive 1. D2:YOURPROG.LIS tells DOS XL to use the file YOURPROG.LIS on drive 2. D:MESSAGE.TXT tells DOS XL to use the file MESSAGE.TXT on drive 1.

Whenever you command DOS XL to do something which causes it to create a new file on a diskette, DOS XL adds the name of that file to something called a "directory". By maintaining this directory, DOS XL knows where each file on your diskette is located just like your address book or telephone directory tells you where all your friends and associates are located. Whenever you tell DOS XL you wish to use one of the files on your diskette, DOS XL reads through this directory, entry by entry, until it finds the entry for the file you requested.

In many cases, you will not wish to specify to DOS XL one specific file; but rather a group of files with similar names. To make this possible, DOS XL allows you to use "wild-card" characters as part of the file name. These wild-card characters will match any other letter or number when DOS XL goes searching through a diskette directory to find the file you have requested.

There are two wild-card characters DOS XL will accept. The first one is a question mark (?) which will match any single letter or number at the same position as the question mark in the file name. For instance, if the files YELL.TXT, BELL.TXT, SELL.BAS, PRELL.TXT, and SELF.TXT were listed in a diskette directory, a wild-card file name of ?ELL.TXT would match YELL.TXT and BELL.TXT.

The second wild-card character is the asterisk or star (*). This character will match a group of zero or more characters. As the following example table shows, this wild-card character can be used alone or combined with the question mark in order to gain a lot of flexibility:

	Wild-Card File Names			
Directory	*ELL.TXT	SEL*.*	?ELL.*	*.*
YELL.TXT	Matches		Matches	Matches
BELL.TXT	Matches		Matches	Matches

SELL.BAS		Matches	Matches	Matches
PRELL.TXT	Matches			Matches
SELF.TXT	Matches	Matches		
SELFISH.COM		Matches		Matches

As you can see, the wild card characters can be used in both the primary file name and the extension. It might appear that wild-card characters are used to tell DOS XL to find the first file which matches the wild-card file name you gave, but actually these characters are much more powerful. Instead of just stopping at the first file DOS XL finds which matches the wild-card file name you specified, DOS XL will continue to perform whatever command you gave it on any other files it can find on that diskette which also match the wild-card file name. This can save you a lot of command typing if you wish to command DOS XL to perform the same command on a large collection of files.

For many of the things you can command DOS XL to do it would not make any sense to specify more than one specific file, and DOS XL won't accept a wild-card file name when you give it these types of commands. The DOS XL menu commands for which it will accept wild-card file names are as follows:

Files on Disk
Copy Files
Erase Files
Protect Files
Unprotect Files

A full discussion of what these commands will cause DOS XL to do is provided later in this manual.

1.4 How Do You Use Your Other Non-Disk Devices?

All Atari Personal Computers consider everything which you have to attach to your computer via a cable (like your Indus GT) to be an "external device" (or "peripheral"). This includes your television set or display monitor. It also includes your computer's keyboard even though it is obviously a direct part of the computer unit. When prompted for a file name by DOS XL, sometimes you can enter the name of one of these other devices instead. These other devices are referred to by names consisting of a single letter optionally followed by a

single digit used to define a specific device when more than one of the same kind exist, such as in the case of D1: and D2: for two diskette drives. The device name must be followed by a colon. The following is a list of device names which DOS XL will recognize:

C: The Program Recorder. This is a device very similar to a standard audio cassette recorder, but which Atari has heavily modified so that your Atari Computer can save and load information to/from it. You can use the recorder as either an input (load or read) or output (save or write) device, but never as both simultaneously.

D1:-D8: Diskette Drive(s). Unlike the program recorder, diskette drives can be used for input and output simultaneously. Since diskette drives can also access any piece of information recorded on a diskette directly without having to "rewind" or "fast-forward" a tape, you usually also have to specify a file name following the device code (D1:-D8:), as previously mentioned, so that DOS XL can tell the diskette drive which piece of information to input or output.

NOTE: If you use D: without a drive number, D1: is assumed.

E: Screen Editor. The screen editor simulates a very limited text editor/word processor using the keyboard as input and the display (television or monitor) as output. This is the editor you use when typing in a BASIC program. In many cases, DOS XL, BASIC, and many other programs will assume that you want to use this device (E:) unless you specify some other device.

K: Keyboard. This device code allows you to specify just the keyboard half of the full screen editor device. This device can be used for input only.

P: Printer. This device code allows you to send text to your printer if you have one. On old Atari computer configurations, output to the P: device was sent to a printer via Atari's 850 Interface Module. Atari's newer printers can be connected directly to your Atari computer without the added expense of this interface module.

R1:-R4: RS-232 Serial Ports. In addition to connecting a printer to an Atari computer, Atari's 850 Interface Module also permits your Atari computer to perform RS-232 communications. RS-232 is an extremely industry standard type

of communications which permits just about any computer to talk to just about any other computer as well as an extremely large number of non-Atari peripherals like video terminals, non-Atari printers, plotters, and modems. These four RS-232 devices can be used for both input and output.

NOTE: If you use R: without a device number, R1: is assumed.

S: Screen Display (either television or monitor). This device code allows you to specify just the screen display half of the screen editor device. Both input and output can be performed to/from this device, thus permitting programs to both change the screen display and read what the screen is currently displaying.

1.5 How Does DOS XL Accept Commands?

A primary feature of DOS XL is the two different ways by which you can command it. You may choose either a menu mode or a Command Processor. In case you are unfamiliar with other menu driven programs such as Atari's own DOS, a menu is simply a list of commands which appear on the screen. You need simply choose one of the options listed before you. If additional information is required, you are further prompted by the program (DOS XL in this case) to type the necessary information. In this way, you need not remember all the special command words which DOS XL will understand; instead, you may simply select a command from the list.

The other way of commanding DOS XL is under DOS XL's Command Processor mode. In this mode, you are not shown a list of commands to choose from. Instead, you must remember (or look up in this manual) all the command words necessary in order to get DOS XL to do your bidding. Although this might at first be cumbersome, once several commands have been committed to memory, the command mode is much faster and easier to use. Also, certain advanced features of the DOS are available only from the Command Processor. Since using DOS XL from under the Command Processor mode is obviously for advanced Atari users, a discussion of its operation has been provided in the later sections of this manual.

The copy of DOS XL you have received with your Indus GT is set up so that you will be presented with the menu mode of command entry. It is recommended that

this mode be used exclusively until you have gained a lot of familiarity with DOS XL

Section 2: GETTING STARTED WITH DOS XL

2.1 How Do You Get DOS XL Running?

"Getting DOS XL Running" is referred to as "booting" DOS XL. "Booting" refers to how every DOS must "pull itself up by its boot straps", which in turn refers to how a DOS must essentially get itself running with only a bare minimum of assistance from the computer itself. After all, every other program you'll run has some sort of DOS (you may not see it) which loads the program into the computer's memory and starts it running; but the DOS itself has no one to help it.

Anyway, here's the step by step approach to booting DOS XL:

1) Connect your disk drive and any other peripherals to your Atari computer following the manufacturer's instructions for each peripheral.

2) Turn on your peripherals and your monitor or television, but leave your Atari computer turned off. **Important reminder:** Never leave any diskette inside any diskette drive when you are turning the drive's power on or off. All drives pass through a sort of "undefined" state when they are being granted or deprived of power. Very extensive circuitry has been built into your Indus GT to prevent it from altering information on your diskettes should they be left in the drive during this "undefined" state (after all, you have no control over a power black-out), but it is always safer to remove any diskettes first.

3) Insert the DOS XL diskette which came with your Indus GT into your disk drive (drive 1 if you have more than one drive). For both Indus and Atari drives, the label of the diskette you want to boot should be facing up and the oval cut which exposes the diskette's surface should be inserted first.

4) For the time being, make sure there are no cartridges inserted into your Atari computer (this includes any BASIC cartridge). It will probably be very common for you to boot DOS XL with cartridges installed (especially a BASIC cartridge) at a later time, but for now you'll want to keep things very simple.

5) Turn on your Atari computer and the rest is automatic: the disk drive will

start doing things, and after a time the DOS XL copyright message will appear at the top of the screen. Then, a message will begin to appear, line by line, which begins with "Welcome to DOS XL...". This message is coming from something called a "start-up file". Getting rid of this start-up file is discussed later in this manual. The last line from the start-up file is just "MENU". This is a command which instructs DOS XL to load the DOS XL menu. After a few more seconds, the menu program will finish loading, the screen will again clear, and the DOS XL menu will appear.

2.2 What About Double Density?

Once you have your DOS XL diskette booted, you will probably start asking yourself about the Indus GT's double density capability. But before starting to use double density, you will need to understand some differences between using single density versus double density. In truth, you will probably find occasion to use both densities. There are many various reasons for using one density versus the other, here are just a few:

Single Density: To be compatible with diskettes created or designed for Atari 810 or 1050 disk drives, you will need to use single density. Single density diskettes used with DOS XL are completely file compatible with Atari DOS 2.0S. All operations which work with Atari DOS will generally work with DOS XL when used under single density.

Are there exceptions? Yes. Several companies produce self-booting disks (that is, you simply put them in the disk drive and turn on the computer, you don't need DOS XL) which make use of certain strange features available only in Atari's own DOS. There is nothing we can do to make DOS XL compatible with these disks! On the other hand, this is not really a problem, since (as we mentioned) these disks are generally self-booting (implying that they include a copy of Atari DOS on their disk). To use these diskettes, do nothing special. Simply follow their manufacturers' directions.

Double Density: Generally, most programs written with cartridge-based languages will work fine with double density DOS XL. This usually includes programs written in Atari BASIC, PILOT, LOGO, Optimized Systems Software BASIC XL, MAC/65, ACTION!, C/65, and more. Of course, if you yourself have written

the program and have not made assumptions about the size and type of disk, your programs will run correctly also.

What doesn't work in double density? Programs which assume that sectors always contain 128 (or 125) characters. Programs which are self-booting and which have special "protection" schemes which prevent you from copying them. Programs which try to perform some or all of the diskette drive input or output themselves without asking for the DOS's assistance. As mentioned above, though, these programs are usually distributed on self-booting disks, so you need do nothing special with them.

How do you use double density? The first thing you will need to do is create what is called a "Double Density System Master Diskette". This diskette is identical to the single density one we sent you, except it will be double density and have more storage capability. Before you can create this double density diskette, you will need to become more familiar with how to command DOS XL. Therefore, creating the Double Density System Master is explained later is this manual.

SPECIAL NOTE: Under DOS XL, a double density diskette will contain exactly the same number of sectors (units of storage) as a single density diskette. This confuses many beginning users. The difference between double density and single density under DOS XL is the size of the sectors. Double density sectors are twice as large as single density sectors, and therefore programs and other data saved on double density diskettes require fewer sectors of storage than they require on single density diskettes.

2.3 How Do You Use The DOS XL Menu?

When you boot the DOS XL diskette which came with your Indus GT, it will automatically place you (after it does a lot of other stuff) into the DOS XL menu. This menu looks something like this:

```
DOS XL MENU   version 2.20
  copyright (c) 1983  OSS, Inc.

Files on Disk     Protect Files
To Cartridge      Unprotect Files
```

```
Copy Files        Rename File
Duplicate Disk    Save Binary
Erase Files       Load Binary
Initialize Disk   Go to Address
Xtended Command   Quit to DOS XL

Enter your selection.
```

Although the above sample does not show it, the first characters of each of the 12 commands in the list are in inverse video (dark characters on a lighted background). These characters are those which you would type to select a command. For example, to command DOS XL to "Rename File", you would simply type "R", in response to the "Enter your selection" prompt. For most of the menu commands, extra input is required. The menu will prompt you with appropriate messages whenever you are required to input more information, such as a file name.

Whenever you are prompted by the DOS XL menu to enter a filename or a filespec, you don't always have to specify the device name for the drive (D1:-D8:). If you do not specify a device name, drive 1 is assumed (D1:). For example, if you typed "GEORGE" as a filename, the menu would assume that you meant "D1:GEORGE".

PROBLEM PREVENTOR: In general, you may not omit the drive specifier on names entered while using cartridge based products such as Atari BASIC or Optimized Systems Software BASIC XL. Except when working with the menu or DOS XL's Command Processor, you usually need to specify the entire file name (and often must enclose it in quotes, as in Atari BASIC).

2.4 Is Your DOS XL Diskette Okay?

In order to ensure that your DOS XL Master Diskette (the one which came with your Indus GT) is in good condition, you should view the names of the files contained on it. The DOS XL menu of commands should be visible on the screen, and the menu will prompt you with:

```
Enter your selection.
```

The first command in the menu is "Files on disk". This command allows you to view the names of the files on a diskette. First ensure that your DOS XL master diskette is still in drive 1. Then command DOS XL to list the names of the files on your diskette by pressing the "F" key on your keyboard.

The DOS XL menu program will respond to this command with:

Files on disk
Filespec:

In order to look at all files on the disk, simply press the <RETURN> key. (Note: <RETURN> is used through-out this manual to indicate the key labelled "RETURN" and not the individual letters "R", "E", "T", "U", "R", and "N".)

The list of files on your master disk will appear and should look something like:

```
 * DOS      SYS 053
 * DOSXL    SYS 046
 * MENU     COM 025
 * CLRDSK   COM 023
 * COPY     COM 075
 * DO       COM 003
 * DUPDBL   COM 011
 * DUPDSK   COM 011
 * INIT     COM 006
 * INITDBL  COM 023
 * RS232    COM 001
 * SDCOPY   COM 086
 * SYSEQU   ASM 022
 * MEM      LIS 066
 * RS232FIXCOM 002
 * CONFIG   BAS 039
 * GTRPM    COM 045
 * VERIFY   COM 001
 * NOVERIFYCOM 001
 * GTSYNC   COM 027
   STARTUP  EXC 003
122 FREE SECTORS
Hit RETURN for menu
```

Each of the files has a primary name and an extension. For example, the file COPY.COM has the primary name COPY and the extension COM. Note that in the file listing the period is not shown. Instead, there may be one or more spaces between the primary name and the extension. This format appears only in the file listing. You may never use this form for specifying files. If you are using file extensions, you must use a period with no intervening spaces, as in "D:TEST.LIS". File names are explained earlier in this manual.

Notice that all of the files on the master disk except the file "STARTUP.EXC" are preceded by an asterisk in the directory listing. An asterisk preceding a file implies that it has been protected from modification or erasure through the use of the DOS XL menu command "Protect Files". This method can also be used to protect your own files from change or deletion through accidental use of one of the DOS XL commands.

2.5 How Do You Keep Your DOS XL Diskette Safe?

Once you have successfully booted your master disk, you will now need to make a second (or "backup") copy. This permits you to put your DOS XL master diskette away in a safe place and only use the second copy you've made. That way if, for some unexplained reason, your second DOS XL diskette gets magnetically altered and will no longer boot, you can recreate it from your safely stored-away master diskette.

First, for safety's sake, place a write protect tab over the write-enable notch on your DOS XL master disk, if it does not already have one. This will prevent any mistakes you may (or may not) make while using DOS XL from damaging your DOS XL master diskette. Note: Some copies of DOS XL may be distributed on diskettes which don't even have write-enable notches which need to be covered. In these cases, you won't have any notch to cover. Location and usage of the write-enable/protect notch is provided in your Indus GT Drive Manual.

The DOS XL booting process should leave you with the DOS XL menu of commands displayed on your screen. The following prompt will appear with the cursor below the menu:

Enter your selection.

What you do at this point depends upon whether you have one or two (or more) drives. If you have two or more drives, and your second drive is an Indus GT, then you'll need to read the section later in this manual which discusses configuring your Indus GT. The reason for reading this later configuration discussion at this point is because you will need to make sure your second drive is configured for single density. Once you've read that section, you can resume at this point. If you only have one drive then you won't need to be concerned about the configuration discussion. If you have a second drive which is not an Indus GT, an Atari 810, or an Atari 1050, then you should consult that drive's manual to see if the drive is "configurable". If it is, you will need to read the later discussion at this point. If it isn't, then you can just continue on here.

In response to the "Enter your selection" prompt, you'll want to type "D" for "Duplicate Disk". This will cause the DOS XL menu program to ask you:

Double density?

As mentioned before, the DOS XL system master which came with your Indus GT is single density; so you'll want to answer this question with "N<RETURN>" for "No". Later on when you are making other backups you would answer "Y" for "Yes" if the diskette you are copying is in double density.

Either the DUPDSK.COM (short for DUPlicate DiSK) or DUPDBL.COM (short for DUPlicate DouBLe density disk) utility program will be read into memory, depending on whether you answered "N" or "Y" to the last prompt. The next prompt is:

Source disk (1,2,3,4) :

Normally, you will copy from drive 1 so type "1". The DOS XL menu will the prompt you with:

Destination disk (1,2,3,4) :

Your answer to this prompt is dependent upon whether you have one or more

drives. If you only have one drive, then the answer is quite obviously "1". If you have two or more drives, then answer with "2" in order to speed up the copying process. Once you've entered your answer, you will then be asked:

```
Format destination disk (Y/N)?
```

Most blank diskettes are unformatted. That is, they are not yet prepared to hold any information. In order to copy files or diskettes to blank diskettes, they must first be formatted. Therefore, answer this question with "Y<RETURN>" for "Yes". If you know that your destination diskette is already formatted, answer "Y<RETURN>" anyway just to be consistent. At this point, the DUPDSK or DUPDBL utility will instruct you to either:

```
Insert source disk into drive 1
And hit RETURN when ready
```

or:

```
Insert source disk into drive 1
Insert destination disk into drive 2
And hit RETURN when ready
```

depending upon whether you are using one or two drives for the copy.

Your master disk should still be in drive 1 at this point, which is the source diskette and drive. If you are using drive 2 for the copy, insert your blank diskette (or a diskette to be completely recopied thus losing any old information the diskette may have contained) into drive 2. Once you have the correct disk(s) in the correct drive(s), press "<RETURN>".

The busy light on the front of your source disk drive will come on, and the diskette duplication utility will respond by saying:

```
Reading source disk
```

If you are only using one drive for the copy, when the busy light on your drive goes out you will be prompted with:

```
Insert destination disk into drive 1
```

And hit RETURN when ready

The diskette duplication utility has read as much as possible of the source disk into your computer. At this time, remove your DOS XL master disk from drive 1 and insert a blank diskette or one which you no longer need the old information on it any longer. When the diskette is in, press "<RETURN>".

The program will respond:

Formatting destination disk

and, after a while,

Writing destination disk

Most of the time, the total information on a diskette is too large to hold in your Atari's memory at one time. This is the case for your DOS XL disk. Therefore, if you are only using a single drive to make the copy, you will be prompted to repeatedly insert your source and destination disks (the DOS XL master and the blank diskette, respectively) until the duplication is complete. Follow these prompts carefully until the diskette duplication utility responds:

Copy same disk again (Y/N)?

This means you, with DOS XL's help, have successfully created a backup copy of your DOS XL master diskette. Remove this backup DOS XL from the drive and place a write-protect tab over the diskette write-enable notch. You should also make a label for the diskette which indicates that it is your single density DOS XL System Diskette ("System Master Diskette" refers to the diskette you received with your drive.)

To get back to the main DOS XL menu, type "N<RETURN>" in response to the above question. The disk duplication utility program will respond with:

Hit RETURN for menu

After you press "<RETURN>", DOS XL will return you to the main command menu.

2.6 How Do You Use Cartridge BASIC or Other Cartridges?

How various cartridges work with DOS XL depends upon which particular cartridge is in use. All cartridges will work with DOS XL exactly the way they work with standard Atari DOS 2.0S. Certain cartridges, especially game cartridges, are designed not to work at all with any DOS. These types of cartridges are written in such a way that your Atari computer starts them running the minute the computer is turned on, before the computer attempts to start booting the DOS. Because DOS is never booted, the cartridges don't require DOS and therefore there is no "using" DOS XL with these cartridges.

The other main type of cartridge is one which wants to use DOS for saving and loading programs, text, or other information. Most of these types of cartridges are programming language cartridges such as Atari's BASIC or Optimized Systems Software's BASIC XL and ACTION!. These cartridges will permit your DOS XL to be booted into the computer, and then the DOS XL will in turn automatically start the cartridge running (this is called "transferring control (of the computer) to the cartridge"). This means that if you boot DOS XL without Atari's BASIC cartridge in your computer, you end up with either the DOS XL menu or at DOS XL command level. But, if you boot DOS XL with the BASIC cartridge installed, you end up with BASIC's "READY" prompt.

There is a certain case in which DOS XL will not automatically transfer control to one of these cartridges. This is when the booted DOS XL diskette contains a file called "STARTUP.EXC". This is a special automatic command file from which DOS XL will take its commands when DOS XL is finished booting. The STARTUP.EXC file is an advanced feature of DOS XL which is discussed later in this manual.

The DOS XL diskette which came with your Indus GT contains a STARTUP.EXC file which produces a series of single line messages prior to giving you the command menu. If the STARTUP.EXC file is "Erased" (the "erase" menu option is discussed later) from a copy of your DOS XL diskette (never alter your original DOS XL diskette) then those "hello" messages will no longer print and you will be placed directly in the command menu.

Since a STARTUP.EXC file can contain a command which tells DOS XL to transfer control to a cartridge, DOS XL will follow the commands in STARTUP.EXC rather than automatically giving the cartridge control. Since the STARTUP.EXC file on the diskette which came with your Indus GT never tells DOS XL to start the

cartridge running, that copy of DOS XL will never automatically leave you in BASIC or any other cartridge. If you erase the STARTUP.EXC file, then DOS XL will automatically give control to any cartridge present (if the cartridge wishes control).

You can always tell DOS XL to transfer control to a cartridge by specifying the "To Cartridge" command on the DOS XL command menu. This command is discussed later.

Section 3: THE DOS XL MENU

The DOS XL menu was designed to be easy to use while allowing you access to the full power of both your Indus GT diskette drive and your Atari computer.

For those of you who have previously used Atari DOS, here is a summary of the differences between the DOS XL menu and the Atari DOS menu:

1) Loading the DOS menu - The Atari DOS menu must be loaded in from the disk whenever you return to DOS from a cartridge. Since we felt that this process was too slow and cumbersome, we made sure that the DOS XL menu may be kept "resident" (in memory) at all times. This does occupy about 2,000 characters more memory, but this is more than offset by the extra ease-of-use. Once you become comfortable with DOS XL, you can then begin to learn how to issue commands directly to DOS XL without using the menu program; thereby freeing up these 2,000 characters of storage for other uses.

2) Atari DOS supports the use of a special file in which to save user memory while accessing DOS functions (the MEM.SAV file). DOS XL neither supports nor needs this file.

3) The DOS XL menu only requires you to press a single key to access commands, whereas Atari DOS insists that you keep pressing the "<RETURN>" key after each menu selection.

Section 3: THE DOS XL MENU

3.1 What is the DOS XL Menu?

Whenever the menu is entered, the following list of commands will appear on the screen:

```
DOS XL MENU   version 2.20
  copyright (c) 1983  OSS, Inc.

Files on Disk     Protect Files
To Cartridge      Unprotect Files
Copy Files        Rename File
Duplicate Disk    Save Binary
Erase Files       Load Binary
Initialize Disk   Go to Address
Xtended Command   Quit to DOS XL
```

When the DOS XL menu is visible on the screen, you are prompted with:

```
Enter your selection.
```

At that point, you should type the first letter of any of the DOS XL menu commands listed on the screen. If more input is required to complete the command, the menu will prompt you for more information. Unless the command loads a utility program, you may cancel a command at any time by hitting the [ESCape] key on the upper left of your keyboard. If the command loads a utility program (the only commands which do this are "Copy Files", "Duplicate Disk", and "Initialize Disk"), you may have to press the [RESET] key in order to cancel the command.

The following sections describe each menu command in detail. The commands are presented in alphabetical order, not the order in which they appear in the menu.

3.2 How Do You "Copy Files" From the Menu?

The "Copy Files" command menu item allows you to transfer files between diskettes or to different files on the same diskette. This command is most

useful for copying one or a few files from one diskette to another. If you desire to transfer all or most of the files on a single diskette to another diskette, you should use the "Duplicate Disk" command menu item instead, as it will perform the operation much more rapidly.

To use the copy command, select "C" when the menu prompts you for a command selection. At that time, DOS XL will check for the COPY.COM file on the diskette in drive 1. This is the utility program which performs file transfers. If DOS XL does not find the COPY.COM program, you will be prompted to insert your DOS XL master disk as follows:

Copy Files
Insert MASTER disk and hit RETURN

If you receive this prompt, take out any disk in drive 1 and insert any DOS XL system disk which is in the same density as the diskette which was in drive 1 and press the <RETURN> key.

The menu will then prompt:

Copy Files
From file:

At this point, you should respond with a filespec specifying the file or files to be copied (e.g., D:GEORGE, D2:JUNK.LIS, etc.). For example, if you want to transfer the contents of the file "PROG1" on drive 1 to another diskette, you should type "D1:PROG1".

Notice that wild-cards may be used to refer to files using the COPY utility (e.g., TEMP.*, AB??.COM, etc.). As a special case, if you wish to copy all files on a disk to another disk, just use a filespec of Dn:, where n is the source drive number (e.g., D1:). How to use wild-cards is discussed earlier in this manual.

The COPY utility will then prompt:

To file:

You should respond with the destination filespec. In most cases you will want to transfer files from one diskette to another without changing their names.

In this case you may refer to the destination filespec as just Dn:, where n is the destination drive number (e.g., D:, D1:, D2:). In the above example, if you wanted to copy "PROG1" to a different diskette and you own only 1 drive, you should type "D1:".

You will then be asked:

    Single Drive?

If you own only a single drive as in the above example, or if you are performing this copy to another diskette in the same drive,

You type: Y<RETURN>

In any other case,

You type: N<RETURN>

The COPY.COM utility program will then be loaded from the diskette, and you will be prompted:

    Insert disk(s) to be copied
    and hit RETURN when ready

Make sure the diskette containing the file you are copying is in the drive you told the copy utility it would be in. If you are using more than one drive to make the copy, insert your destination disk into the proper drive also.

You type: <RETURN>

Before each file is copied, you will be asked:

    Copy
    Dn:filename
    to Dn:filename?

If you wish to copy that particular file,

You type: Y<RETURN>

Otherwise,

You type: N<RETURN>

If you choose not to copy a file, a message will be printed to the screen verifying that the file was not copied.

Whenever you do choose to copy a file, the source file will be read into memory. If you are copying to another disk on the same drive, you will then be prompted to insert the destination disk as follows:

```
Insert 'to' disk and hit RETURN
```

If the destination file already exists, you will be asked:

```
'To' file already exists
OK to overwrite?
```

If you wish to replace the old file with the source file,

You type: Y<RETURN>

Otherwise,

You type: N<RETURN>

If the destination file has previously been guarded against modification by using the DOS XL menu "Protect Files" command (i.e., the file is preceded by an asterisk in the "Files on Disk" listing of the disk), the COPY utility program will not be able to overwrite that file. The protection must first be removed using the "Unprotect Files" menu item before any writes to that file may occur!

The COPY utility reads as much as possible of the source file into memory at one time. If the source file is too large to fit into memory and you are copying on a single drive, you will again be prompted:

```
Insert 'from' disk and hit RETURN
```

Re-insert your source disk and continue to carefully follow the directions of the prompts until the entire file is copied.

When a file has been completely copied, a verification message will be printed on the screen. When all files have been copied, you will be prompted:

Hit RETURN for menu

To return to the list of menu commands,

You type: <RETURN>

PROBLEM PREVENTOR: The "Copy Files" command should not be used to copy from single to double density diskettes if you own only 1 disk drive. Copying between two different densities using only one drive is discussed as a special case later in this manual.

PROBLEM PREVENTOR: The "Copy Files" options should never be used to copy the file "DOS.SYS" onto another diskette. DOS.SYS is a special file which is handled in a very special way by DOS XL (in order to remain fully compatible with Atari's own DOS). Because of this special handling, using "Copy Files" to copy DOS.SYS will result in the "to" DOS.SYS file containing two copies of the DOS.SYS file; and in many cases the "to" diskette will be unbootable. Only the "Initialize Disk" option should be used to place a copy of DOS.SYS onto a diskette. After DOS.SYS has been placed onto a diskette using "Initialize Disk", an "all files" copy can then be performed (by specifying the "*.*" file name). Since you will already have placed DOS.SYS on the "to" diskette, "Copy Files" will ask if it is "OK to overwrite?" the DOS.SYS file. Simply answer "N" for "no", and "Copy Files" will move on to copying the other files.

PROBLEM PREVENTOR: When using "Copy Files" with only a single drive, it will not bother to ask you to "Insert 'from' disk and hit RETURN" if you answered an "OK to overwrite?" question with "N" for "no". Therefore, always remember to insert your "from" disk before answering this question with"N". "Copy Files" will remember to ask you to "Insert 'from' disk and hit RETURN" after it is done copying the file if you answered "OK to overwrite?" with "Y" for "yes".

3.3 How Do You "Duplicate (a) Disk" From the Menu?

The "Duplicate Disk" menu item allows you to quickly copy the entire contents of a diskette to another diskette. If you wish to copy only one or just a few files from one diskette to another, or if you need to preserve some of the files already on the disk you wish to copy to, the "Copy Files" menu item should be used instead.

PROBLEM PREVENTOR: The "Duplicate Disk" menu item writes entirely new information to the destination diskette, thus completely erasing all files which previously existed there. Carefully select the desired destination diskette to avoid accidently destroying your program disks.

To select the "Duplicate Disk" menu item, type "D" when you are prompted to enter a command selection.

You will then be asked:

Duplicate Disk
Double density?

If your source disk was formatted under single density,

You type: N<RETURN>

If the source disk is double density,

You type: Y<RETURN>

After you answer this question, DOS XL will check for the DUPDSK.COM (for single density) or DUPDBL.COM (for double density) file on the diskette in drive 1. This is the utility program which performs full diskette duplication. If DOS XL does not find the utility program it needs, you will be prompted to insert your DOS XL master disk as follows:

Duplicate Disk
Insert MASTER disk and hit RETURN

If you receive this prompt, take out any disk in drive 1 and insert any DOS XL system disk which is in the same density as the diskette which was in drive 1 and press the <RETURN> key.

Once loaded, the diskette duplication utility will prompt you with:

    Source disk (1,2,3,4) :

Normally, you will copy from drive 1 so,

You type: 1<RETURN>

Next you will be prompted with:

    Destination disk (1,2,3,4) :

If you have only a single drive, or you wish to use drive 1 as your destination drive,

You type: 1<RETURN>

If you wish to use a drive other than 1 for a destination drive,

You type: n<RETURN>

where "n" is the number of the desired destination drive.

Once you've specified the destination drive, you will be asked:

    Format destination disk (Y/N)?

Most blank diskettes are unformatted. That is, they are not yet prepared to hold disk files. In order to copy files or diskettes to blank diskettes, they must first be formatted. Therefore,

You type: Y<RETURN>

At this point, the disk duplication utility will instruct you to either:

```
Insert source disk into drive 1
And hit RETURN when ready
```

or:

```
Insert source disk into drive 1
Insert destination disk into drive n
And hit RETURN when ready
```

In either case, follow the directions given by the utility. Once you have the diskette(s) inserted,

You type: <RETURN>

At this point, the light on the front of your disk drive will come on, and the disk duplication utility will respond by saying:

```
Reading source disk
```

If the destination drive is the same as the source drive, you will be prompted:

```
Insert destination disk into drive n
And hit RETURN when ready
```

At this time, remove your source diskette from drive 1 and insert a blank diskette. When this has been done,

You type: <RETURN>

The program will respond:

```
Formatting destination disk
```

and, after a while,

Writing destination disk

Most of the time, the total information on a diskette is too large to hold in your Atari's memory at one time. This is the case for your DOS XL disk. Therefore, you will be prompted to repeatedly insert your source and destination disks until the duplication is complete. Follow these prompts carefully until the disk duplication utility responds:

Copy same disk again (Y/N)?

You type: N<RETURN>

At this point, you will be prompted:

Hit RETURN for menu

To return to the DOS XL menu,

You type: <RETURN>

3.4 How Do You "Erase Files" From the Menu?

The "Erase Files" menu item allows you to delete one or more files from a diskette. This menu item should be used with care since only very advanced users can recover an erased file (if at all).

If you use the "Erase Files" menu item to attempt to erase a file which has previously been protected (i.e., the file name is preceded by an asterisk in the directory listing), you will be given the error message "FILE PROTECTED". If you desire to erase this file, you must first remove the protection by using the "Unprotect Files" command. Note that protecting files is an excellent way of guarding against accidental erasure.

To use this command, select "E" when the menu prompts you for a command selection. The menu will then prompt:

Erase Files

Filespec to erase:

You should respond with the name of the file you wish to erase. If you wish to erase a group of files, you may use wild-card characters in the filespec. However, be very sure you know what you are erasing. Wild-card characters are discussed earlier in this manual.

You will then be asked:

Are you sure?

If you feel the filespec you entered was correct,

You type: Y<RETURN>

If you wish to abort the "Erase Files" command,

You type: N<RETURN>

If you answered "Y", all files which match the selected filespec will be removed from the diskette. The menu will then prompt:

Hit RETURN for menu

To return to the menu of commands,

You type: <RETURN>

3.5 How Do You List The "Files On (a) Disk" From the Menu?

The "Files on Disk" menu item allows you to view the names of any or all files on a diskette.

To use this command simply select "F" when prompted by the menu for a command selection. Then insert the desired diskette into one of your disk drives (or drive 1 if you have only one drive). At that point, the menu will prompt:

Files on disk

Filespec:

The requested filespec instructs DOS XL which files on the disk to look for and display. The following table gives some examples of filespecs and the corresponding lists of files they display:

Filespec	Files listed
GEORGE	The file having the name GEORGE, if such a file exists.
JUNK.SAV	The file having the primary name JUNK and the extender SAV, if such a file exists.
AB?	Any file not having an extender whose name is three characters long where the first two are AB. This filespec matches ABC, ABX, AB1, etc.
CAT*	Any file whose name begins with CAT. The filespec matches CAT, CATCHER, CATTLE, etc.
JOHN.??X	Any file whose primary name is JOHN and whose extender is three characters long ending in X. This matches JOHN.ABX, JOHN.XXX, etc.
.	All files on the diskette. This filespec may be abbreviated by just <RETURN>.
D1:	All files on the diskette in drive 1.
D2:	All files on the diskette in drive 2.

NOTE: After entering the filespec you desire, you can optionally enter a space and request an "output" specifier -- similar to the intrinsic command processor level DIRectory command discussed later in this manual. For instance, entering:

Files on disk
Filespec: D1: P:

would result in a directory of all of the files on drive one being sent to your printer.

After entering the filespec, press <RETURN>. DOS XL will then give you a listing of all the files on the diskette which match your filespec. If an asterisk (*) prints in front of the file name on the listing, then the file is protected from being erased or altered. The number following the file name is the number of sectors on the diskette which the file occupies. A sector can contain about 125 characters in single density, and about 253 in double density.

Following this listing, you will get:

    Hit RETURN for menu

So, just type <RETURN> to return to the menu.

3.6 How Do You "Go to (an) Address" From the Menu?

The menu item "Go to Address" is normally only used by very advanced Atari users. It allows you to command DOS XL to pass control of your computer to a machine language program already residing in your Atari computer's memory. This program should have previously been loaded into memory using the DOS XL menu "Load File" command, or an equivalent method.

To use the "Go to Address" command, type 'G' when the menu prompt "Enter your selection." appears. At that time, the menu will prompt:

    Go to Address
    Address:

You should respond with the hexadecimal address of the location in memory to which DOS XL is to transfer control (to which it is to jump). For example, if a machine language program resides at location $5000 (the dollar sign indicates hexadecimal), you would respond with "5000". Note that although the number is a hexadecimal value, you should not precede it with a dollar sign

when you enter it.

Be sure that the address you enter is correct; for, in general, if you pass control to a location in memory which does not contain the desired machine language program, control of your computer will be lost and the keyboard will "hang" (not respond to you). In some cases, hitting the [RESET] key on your computer's keyboard will return control to you. Most of the time, however, you will be forced to turn off the power to your computer and repeat the boot process.

Once you press <RETURN> following the address, control will be passed to the machine language routine located at the desired address. If that routine returns to the menu with a 6502 RTS instruction, you will be instructed to:

Hit RETURN for menu

To return to the menu of commands,

You type: <RETURN>

NOTE: This menu item is strictly for very advanced Atari users. If you don't understand some of the terms and procedures discussed for using it, don't worry. That probably just means you don't have any need to use it.

3.7 How Do You "Initialize (a) Disk" From the Menu?

The "Initialize Disk" command allows you to format blank diskettes so that you may use them to store program and data files. If you wish to create a bootable diskette rather than just a data diskette, you will normally want to duplicate your DOS XL master disk. In this case you should use the "Duplicate Disk" command rather than the "Initialize Disk" command. If you want to duplicate any of your diskettes using the "Duplicate Disk" command, you do not need to format them first using the "Initialize Disk" command, for the "Duplicate Disk" utility will perform the format operation if you desire.

PROBLEM PREVENTOR: The "Initialize Disk" command writes entirely new information to the desired diskette, thus erasing completely all files which previously existed there. Carefully select the diskette to initialize to avoid

accidently destroying your program disks.

To use this command, select "I" when prompted for a command selection. At that point, DOS XL will check for the presence of the INIT.COM utility on the diskette in drive 1. If it is not there, you will be prompted:

Initialize Disk
Insert MASTER disk and hit RETURN

after which you should insert a DOS XL system disk which is of the same density as the diskette you remove from drive 1. Then hit the <RETURN> key.

The INIT utility program will be loaded into memory and you will be presented with the four options of the INIT program. They are:

1. Format disk only
2. Format disk and write DOS.SYS
3. Write DOS.SYS only
4. Reformat boot tracks only
5. Exit to DOS XL

PROBLEM PREVENTOR: The "I" (INIT) option may normally be used only to initialize diskettes of the same density as the operating density of the drive used for the initialization.

PROBLEM PREVENTOR: Normally, you should use the "Duplicate Disk" command to copy your DOS XL master to create a spare bootable disk. However, option 2 of the "Initialize Disk" command may be used to create a bootable disk. Do not, however, use option 2 if you have booted the system with an Optimized Systems Software SuperCartridge inserted. In that configuration, DOS XL will not properly write the DOS.SYS file. In order to use option 2 of the "Initialize Disk" command, first remove the SuperCartridge and then reboot the system. After using option 2 to create a bootable disk, the DOSXL.SYS file should be copied onto that disk if it is to be used with a SuperCartridge.

PROBLEM PREVENTOR: You must not be using an extended memory copy of DOS XL while initializing new diskettes. You must use the standard version of DOS XL. (See discussion of the DOSXL.XL and DOSXL.SUP files elsewhere in this manual.) If you are using the OSS Supercartridge version of DOS XL (DOSXL.XL), then all you need to do is power off your computer, remove any

Supercartridge, and then power your computer back on to reboot the standard DOS XL version. If you have a 64K Atari computer ("XL" series computer) and have renamed DOSXL.XL to DOSXL.SYS in order to use the extended memory, you must rename DOSXL.SYS to some other name (normally just DOSXL without the SYS extension), and then turn you computer off and then on again to reboot the standard DOS XL version. (Note: extended memory DOS XL users who use the DOSXL.XL copy of DOS may find it easier to simply keep some DOS system diskettes laying around which are not set up to use the extended memory, that way you can simply boot these diskettes instead of always renaming DOSXL.SYS to something else.)

PROBLEM PREVENTOR: When you use option 2 of the initialization menu, the copy of DOS XL which will be written to your newly initialized diskette will expect to find the menu program (MENU.COM) when the new diskette is booted. But, the initialization utility does not place MENU.COM on your newly initialized diskette for you. In order to boot this new diskette, you must first use the "Copy Files" command menu item to copy MENU.COM from one of your DOS XL system diskettes onto your newly initialized diskette. If you forget to do this, the new DOS XL diskette will start to boot but will "die" (stop working) before it says anything on the screen.

PROBLEM PREVENTOR: Option 4 of the initialization menu has to do with special formatting required for optimum performance under synchromesh. Specifying option 4 will destroy any existing data on your diskette. Refer to Section 14 of this manual for a complete discussion of synchromesh and its initialization requirements.

If you wish to create a bootable disk,

You type: 2<RETURN>

If you wish to create just a data disk,

You type: 1<RETURN>

Once you've selected your initialization option, you will be asked:

```
Drive (1,2,3,4):
```

You should respond with the desired drive number (always 1, if you have only one drive), followed by <RETURN>.

After selecting the drive to use for the initialization, you will be asked:

Option n drive n - Are you sure (Y/N)?

If you are happy with your entries so far,

You type: Y<RETURN>

Otherwise,

You type: N<RETURN>

If you typed "Y", the specified command will be executed. Once the initialization is completed, the initialization utility will prompt you with:

HIT RETURN FOR NEW FUNCTION

Once you press <RETURN>, you will again be presented with the four options.

If you have more disks to initialize, repeat the above steps. Otherwise,

You type: 5<RETURN>

and you will be returned to the DOS XL menu.

3.8 How Do You "Load (a) Binary" File From the Menu?

The "Load Binary" menu item allows you to read a binary file from disk into the memory of your Atari computer. This command can be used to load binary object of assembly language programs, or binary data to be used by such programs. The file you wish to load should have previously been written to disk using the DOS XL menu "Save File" command, or an equivalent method.

PROBLEM PREVENTOR: Do not use this command to load Atari BASIC or BASIC XL programs into memory. Instead, just use the LOAD command from the BASIC cartridge (i.e., after you have been given the "READY" prompt).

To use the "Load Binary" command, type "L" when prompted to enter your command selection. The menu will then prompt:

Load Binary
Filename:

You should respond with the name of the previously saved file you wish to load. For example, if you wish to load into memory the file "FILE1.OBJ" on drive 1, you should type "D:FILE1.OBJ".

At this point, DOS XL will access the disk to read in the binary file. You will then be asked:

Hit RETURN for menu

To return to the DOS XL menu of commands,

You type: <RETURN>

NOTE: This menu item is intended for the very advanced Atari user. If you don't fully understand this discussion, don't worry about it. That probably just means you have no need to use this menu item.

3.9 How Do You "Protect Files" From the Menu?

In many cases you will have created files on your disks which you know you will hardly ever need to modify. There is a way to guard these files so that you need not worry about accidently deleting or modifying their contents. The "Protect Files" command allows you to protect files from renaming, erasure, or modification. These files will then be preceded by an asterisk in a directory listing when you use the "Files on Disk" command. If in the future you desire to remove the protection afforded by this command, you should use the "Unprotect Files" command.

To use this command, select "P" when the menu prompts you for a command selection. The menu will then prompt:

Protect Files
Filespec to protect:

You should respond with the name of the file you wish to protect. If you wish to protect a group of files, you may use wild-card characters in the filespec. Wild-card characters are discussed earlier in this manual.

At this point the disk will be accessed and the files will be protected. The menu will then prompt:

Hit RETURN for menu

To return to the menu of commands,

You type: <RETURN>

3.10 How Do You "Quit to DOS XL" Command Level From the Menu?

The "Quit to DOS XL" command is used to pass control from the DOS XL menu to the DOS XL Command Processor. Although almost all the functions you need from DOS may be accomplished from the DOS XL menu, certain advanced commands and features are accessible only from the Command Processor mode.

To use this command, type "Q" when prompted by the menu to enter a command selection. At that point, control will be transfered to the Command Processor mode. In place of the menu, the prompt "D1:" will appear at the upper left corner of the screen. A short discussion of Command Processor mode (or command level) is provided later in this manual.

3.11 How Do You "Rename (a) File" From the Menu?

The "Rename File" command may be used to change the file name associated with

a file of information. This command does not alter or delete any information contained in the file. Rather, the file will only show up with a different name in the directory listing when using the "Files on Disk" command.

PROBLEM PREVENTOR: If you attempt to rename a file which has been protected against modification (i.e., the file name is preceded by an asterisk in the directory listing), you will be given the error message "FILE PROTECTED". If you desire to rename this file, you must first remove the protection by using the DOS XL menu command, "Unprotect Files".

To use the "Rename File" menu item, select "R" when the menu prompt, "Enter your selection." appears. You will then be asked:

Rename File
Old name:

You should respond with the current name of the file who's name you wish to change. For example, if you want to change the name of the file "D:GEORGE" to "D:PROG1", you should type "D:GEORGE".

The menu will then respond,

New name:

At this point you should type the new name you wish the file to have. In the above example, you should type "PROG1" at this time. Notice that you must not use a device specifier (i.e., D:, D2:, etc.) in the new name; you should type just "PROG1", not "D:PROG1".

You will then be asked,

Are you sure?

If you are satisfied that you have entered both file names correctly,

You type: Y<RETURN>

If instead you wish to abort the rename operation,

You type: N<RETURN>

If you answered with "Y", the designated file will be renamed, and you will be prompted:

Hit RETURN for menu

To return to the list of menu commands,

You type: <RETURN>

3.12 How Do You "Save (a) Binary File" From the Menu?

The "Save Binary" command allows you to write a portion of your Atari computer's memory to a disk file. This command can be used to save to disk binary object of assembly language programs, or binary data to be used by such programs.

PROBLEM PREVENTOR: Do not use this command to save Atari BASIC or BASIC XL programs from memory. Instead, just use the SAVE command from the BASIC cartridge (i.e., after you have been given the "READY" prompt).

PROBLEM PREVENTOR: If you attempt to save binary data to a file which has been protected against modification (i.e., the file name is preceded by an asterisk in the directory listing), you will be given the error message "FILE PROTECTED". If you desire to rename this file, you must first remove the protection by using the DOS XL menu command, "Unprotect Files".

To use the "Save Binary" command, type "S" when prompted to enter your command selection. The menu will then prompt:

Save Binary
Filename:

You should respond with the name you wish the saved file to have. For example, if you wish to write memory from locations $4000 to $4100 (the dollar signs indicate hexadecimal addresses) to the file "FILE1.OBJ" on drive 1, you should type "D:FILE1.OBJ".

It is recommended that binary object file names have either the extension "OBJ", or "COM". In the former case, "OBJ" would indicate that the file was an assembly language OBJect file for a program or data. The second extension, "COM", indicates that the program is a system utility program which was either included with your DOS XL master disk or written by you or another user.

Once you've specified the file name and press <RETURN>, you will be prompted with:

    Starting address:

You should respond with the hexadecimal value of the first address you wish to write to disk. In the above example, the starting address was $4000 so you should type "4000". Note that although the value is hexadecimal, you should not precede the number with a dollar sign.

The menu will then prompt with:

    Ending address:

You should respond with the hexadecimal value of the last address you wish to save. In the previous example, you should enter "4100".

At this point, DOS XL will access the disk to write out the binary file. You will then be instructed to:

    Hit RETURN for menu

To return to the DOS XL menu of commands,

You type: <RETURN>

NOTE: This menu item is intended for use by very advanced Atari users. If you don't understand parts of the discussion of this menu item don't worry. Chances are you won't need to use this menu item anyway.

3.13 How Do You Go "To (a) Cartridge" From DOS XL?

The menu item "To Cartridge" permits you to instruct DOS XL to start running (transfer control to) an installed cartridge. (BASIC, for instance.)

To use this command, select "T" when prompted by the menu for a command selection. At that time, DOS XL will transfer control to any cartridge you have installed in your Atari. If the installed cartridge is Atari BASIC, you will then see BASIC's "READY" prompt. If you are using some other cartridge, you should see whatever prompt that cartridge produces. If no cartridge was inserted, the error message "NO CARTRIDGE" will be displayed.

If the "To Cartridge" command is used after any of the following commands are selected:

Copy Files
Duplicate Diskette
Initialize Diskette
Xtended Command
Load Binary

a coldstart will be performed by the cartridge, thus erasing any program which was in memory. Therefore, if you wish to go to the menu to execute any of these commands, remember to first write any program you are working on to disk. This is accomplished in Atari BASIC or Optimized Systems Software's BASIC XL by using the SAVE command in the BASIC cartridge.

3.14 How Do You "Unprotect Files" From the Menu?

The "Unprotect Files" menu item removes the protection status placed on a file by the "Protect Files" menu item. This in turn allows these files be renamed, erased, or modified. These files will no longer then be preceded by an asterisk in a directory listing when you use the "Files on Disk" command.

To use this command, select "U" when the menu prompts you for a command selection. The menu will then prompt with:

Unprotect Files

Filespec to unprotect:

You should respond with the name of the file you wish to unprotect. If you wish to unprotect a group of files, you may use wild-card characters in the filespec. Wild-card characters are discussed earlier in this manual.

At this point the disk will be accessed and the files will be unprotected. The menu will then prompt with:

Hit RETURN for menu

To return to the menu of commands,

You type: <RETURN>

3.15 How Do You Perform An "Extended Command" From the Menu?

This command may be used to pass a command line to the DOS XL Command Processor. Although almost all the functions you need from DOS may be accomplished from the DOS XL menu, certain advanced commands and features are accessible only from the Command Processor mode. The "Xtended Command" function of the DOS XL menu may be used to access from the menu those commands available only from the Command Processor. The Command Processor's commands are briefly discussed later in this manual.

To use the "Xtended Command" function, select "X" when prompted by the menu "Enter your selection.". At that time, the menu will prompt with:

Xtended Command
Command:

You should respond with the DOS XL command you wish to have executed. For example, if you wish to use the "RS232" command, you should type "RS232".

PROBLEM PREVENTOR: Many of the DOS XL commands accessible by the "Xtended Command" function perform their operations by loading utility programs from a DOS XL system disk. If you wish to use a command which employs a utility program (one of the ".COM" files on a DOS XL system diskette), you should

insure that a DOS XL system disk is first inserted into drive 1.

Once you press <RETURN> after entering the extended command, the desired command will be passed to the DOS XL Command Processor and executed. When the command is finished executing, the menu will prompt you with:

    Hit RETURN for menu

To return to the DOS XL menu of commands,

You type: <RETURN>

PROBLEM PREVENTOR: The following commands should not be used from the menu via the "X" command:

    CONFIG   SDCOPY   INITDBL

These commands alter the density selection(s) for the disk drives and thus alter the amount of memory used for disk buffers. To be safe, always use them only from the Command Processor Level.

Section 4: THE DOS XL COMMAND PROCESSOR

4.1 Where Does The "Quit To DOS XL" Menu Item Leave You?

The DOS XL menu program is actually just a very special utility program which has been added to the DOS XL operating system itself. DOS XL normally likes to accept the commands you give it through its "Command Processor". This is a very fancy name for a piece of DOS XL which interprets the English-like commands you type into some action which DOS XL is to perform for you.

The menu utility program is really nothing more than a helper for the beginning DOS XL user. All it does is translate the little single letter menu selection you make into the harder to remember English-like commands, and then sends these commands to the DOS XL Command Processor for the commands to be acted upon.

When you specify "Quit to DOS XL" when using the DOS XL command menu program, the menu program essentially just stops running by transferring control of your Atari directly to the DOS XL Command Processor. When you are using the Command Processor rather than the menu, you are said to be at "Command Level". You can always tell when you are talking (using your Atari keyboard) to the DOS XL Command Processor, because the Command Processor will prompt you with:

```
D1:
```

This "D1:", which appears to the left of the cursor, serves two purposes. First, it is the Command Processor's prompt for you to enter a new command. Second, it indicates that drive one (D1:) is currently the "default" drive. The default drive is the one which DOS XL will use whenever you provide a filespec which doesn't explicitly give a disk drive device specifier. For instance, DOS XL would use the default drive when you use filespecs like GEORGE, HERNY.TXT, MYPROG.BAS. If you explicitly give a drive specification as part of your filespec (D1:GEORGE, D3:HENRY.TXT, D2:MYPROG.BAS), DOS XL will use the drive you explicitly specified. The default drive is also the one DOS XL will go to when you enter a Command Processor command which requires DOS XL to load a special utility program from your system diskette. For instance, the COPY command requires DOS XL to load the COPY.COM program from your currently selected default drive. Changing the default drive is discussed later in this

section.

The way in which you must type any DOS XL command, along with the way you must type any required/optional additional information required by the command (the command parameters), is called the "syntax" of the command. If you do not type the command you wish to issue to DOS XL using the correct "syntax" for that command, DOS XL will either do the wrong thing (if it thinks you meant something else which it did understand), or it will give you an error message telling you it didn't understand what you wanted.

In the descriptions of the commands below, special symbols are used to summarize each command's syntax. These symbols have the following meaning:

<> Angle Brackets: These are used to surround the description of a parameter. They are used to indicate that you are not supposed to type what appears between them as part of the command, but rather to substitute a file's name (<file-name>), a drive identifier (<drive-id>), or some other piece of information in place of the <token>. The <token>s are simply place holders for some other varying piece of information which you must supply. Exactly what you should substitute for each <token> depends upon the command you are issuing, and therefore each <token> you should give with a command is detailed with the discussion of that command.

<RETURN> This is a special <token> which is used to represent the key labelled "RETURN" on your Atari keyboard.

[] Square Brackets: These are used to surround the sections of a command which are optional and therefore need not be typed. Square brackets are sometimes nested one inside the other. For instance, [[<drive-id>]<file-name>] indicates that the entire thing is optional and therefore need not be typed, or just <file-name> can be typed, or <drive-id> and <filename> can be typed. However, <drive-id> cannot be typed alone.

Lower Case Letters: These are used for <token> names to further emphasize the fact that the <token> itself is not supposed to be typed, but rather some piece of information is supposed to be substituted for the <token>.

Upper Case Letters: Whenever upper case letters appear in a command description, they must be typed exactly as they appear (or not at all if, and only if, they appear as optional within square [] brackets).

Spaces (blanks): Spaces only appear in command descriptions where ever you must type them. Spaces are never placed in a command description simply to enhance readability.

Other characters: All other characters should be treated exactly like upper case letters. They must be typed exactly as they appear (or not at all if, and only if, they appear as optional within square [] brackets). The only exception to this is the hyphen/dash (-) which is often used as part of <token-names>.

4.2 DOS XL Intrinsic Commands

Instrinsic Commands are one class of commands which can be given anytime the Command Processor's prompt (D#:) appears. Intrinsic commands differ from extrinsic commands by the fact that everything needed to process these commands is already contained within the DOS XL which was loaded into memory when you booted your system. DOS XL does not need to go back to your system diskette in order to get more information when you issue an intrinsic command, therefore you don't need to have your DOS XL diskette in the disk drive when you issue one of these commands.

The following is a summary of the intrinsic commands:

```
@<batch-file>
CAR[TRIDGE]
D#:
DIR[ECTORY] [<drive-id>][<file-spec>] <list-spec>
ERA[SE] [<drive-id>][<file-spec>]
LOA[D] [<drive-id>][<file-name>]
PRO[TECT] [<drive-id>][<file-spec>]
REM[ARK] <any-text>
REN[AME] [<drive-id>]<old-name> <new-name>
```

```
RUN [<entry-address>]
SAV[E] [<drive-id>]<file-name> <starting-address> <ending-address>
TYP[E] [<drive-id>]<file-name> <list-spec>
UNP[ROTECT] [<drive-id>]<filespec>
```

Note that all intrinsic commands may be abbreviated to their first three characters. As a matter of fact, DOS XL only looks at the first three characters while testing for an intrinsic command. Each of the commands will be covered in detail later in this manual; however, to give you a feel of the intrinsic commands, let's look at the DIRECTORY command. While looking at these examples, assume the "D1:" at the beginning of each line is the default drive and has been displayed on the screen by the Command Processor.

```
D1:DIRECTORY        list names of all files on disk in drive one
D1:DIRTY             "     "   "   "    "    "   "   "    "    "
D1:DIR               "     "   "   "    "    "   "   "    "    "
D1:DIR *.*           "     "   "   "    "    "   "   "    "    "
D1:DIR D1:           "     "   "   "    "    "   "   "    "    "
D1:DIR D1:*.*        "     "   "   "    "    "   "   "    "    "
D1 DIR D2:           "     "   "   "    "    "   "   "    "   two
D1:DIR D2:*.*        "     "   "   "    "    "   "   "    "    "
D1:DIR *.OBJ        list names of all files with .OBJ extension
D1:DIR D2:*.ASM     list names of all files with .ASM extension
                      on drive two
```

4.2.1 How Do You Execute a Batch File From Command Level

Command Syntax:

```
@<batch-file>
```

Command Token(s):

<batch-file>

This is the name of the file containing the batch commands for DOS XL to being executing. Only the primary part of the file name can be given, DOS XL assumes the full name to be "<batch-file>.EXC".

Command Option(s):

None

Description:

The @ command tells DOS XL to begin taking commands from a batch file. This file is a text file which may contain both intrinsic and extrinsic DOS XL commands.

The "END" command is used within the batch file to tell DOS XL to stop taking its commands from the file and return to taking commands from the keyboard. In addition, the "CAR"tridge command will cause DOS XL to return to taking commands from the keyboard after the "CAR"tridge command has been executed.

See "Controlling DOS XL Without Being There" for more details.

Example(s):

Suppose the file TEST.EXC contains the following commands:

```
DIR D:<RETURN>
DIR D2:<RETURN>
END<RETURN>
```

Issuing the command:

```
@TEST<RETURN>
```

would tell DOS XL to start taking commands from the file "TEST.EXC". This would cause DOS XL to give a directory listing of the files on the diskette in drive one, followed by a listing of files on the diskette in drive two.

4.2.2 How Do You Go To a CARtridge From Command Level?

Command Syntax:

```
CAR[TRIDGE]
```

Command Token(s):

None

Command Option(s):
None

Description:

The "CAR"tridge command allows the user to enter a cartridge from the command level of DOS XL. The cartridge will retain control of your Atari computer until a cartridge dependent command is given telling the cartridge program to return control to DOS XL. (For the Atari BASIC cartridge, this command is "DOS".)

SPECIAL NOTE: Some cartridges will not return control to DOS, normally because they are not intended to be used together with DOS. Most game cartridges are designed this way.

If no cartridge is present, DOS XL will respond with an error message.

Example(s):

Given that the Atari BASIC cartridge is installed, typing:

CAR<RETURN>

will start BASIC running.

4.2.3 How Do You Change The Default Drive From Command Level?

Command Syntax:
D<1-4>:

Command Token(s):
<1-4>
This indicates a single digit from "1" to "4" which indicates which drive is to become the new default drive.

Command Option(s):

None

Description:

Whenever the DOS XL's Command Processor is ready to accept a command from you, it prompts you with "D#:". This prompt serves a secondary purpose: it reminds you which disk drive is currently selected as the default drive.

Anytime you specify a filename to Command Processor (as part of either an intrinsic or extrinsic command), if you omit the disk specifier, the Command Processor prefixes the filename with the same three characters with which it prompted you. For example, when DOS XL is first booted, the current default drive is drive one, the Command Processor is prompting you with "D1:"; and the Command Processor would see a command like "PROTECT MYFILE.TXT" as "PRO D1:MYFILE.TXT".

If you have more than one drive, however, it is sometimes convenient to designate a drive other than drive one as the default disk drive. So, if you type "D2:" in response to the DOS XL prompt, the Command Processor will begin displaying a new prompt, "D2:", and any files given without a drive designator will be presumed to be on drive two.

Note that DOS XL actually supports D1: through D8:, but most standard drives may only be addressed as D1: through D4:.

SPECIAL NOTE: Unless DOS XL is modified as discussed under "What Do You Do If You Have More Than Two Drives?", it will only accept drives D1: and D2:.

Example(s):

Given that the Command Processor is currently prompting you with:

D1:

if you typed:

```
DIR<RETURN>
```

you would then get a directory listing for the diskette in drive one. If you then typed:

```
D2:<RETURN>
```

the Command Processor would begin prompting you with:

```
D2:
```

and if you then typed:

```
DIR<RETURN>
```

you would get a directory listing for the diskette in drive two since that would now be your default drive. Finally, typing:

```
D1:<RETURN>
```

would restore your default drive to being drive one.

4.2.4 How Do You Get a DIRectory listing From Command Level?

Command Syntax:

DIR[ECTORY] [<drive-id>][<file-spec>] [<list-spec>]

Command Token(s):

<drive-id>

This is an optional parameter which, if used, tells DOS XL to list the directory of files from the specified drive rather than the default drive. If <drive-id> is not included as part of the command, DOS XL will assume the default drive.

<file-spec>

This is an optional parameter which, if used, tells DOS XL to limit the directory display to just those files which meet the <file-spec>. Wildcard characters can be included as part of <file-spec>

(see "So What is a Diskette File Anyway?" for discussion of wildcard characters), thus permitting DIRectory listings of certain groups of files.

<list-spec>
This is an optional parameter which, if used, tells DOS XL where to send the directory listing instead of the screen. See Problem Preventor note below.

Command Option(s):
None

Description:

The DIRectory command searches the disk directory of the specified disk (or the current default drive, if <drive-id> is omitted) for all files matching <file-spec>. The names of all files matching the specifier are then printed to the screen, together with the length of the file (in sectors). An asterisk preceding the file's name indicates that the file is protected from erasure, writing, or renaming.

If <list-spec> is included as part of the command, the directory listing will be sent to that file or device instead of to the screen.

PROBLEM PREVENTOR: If <list-spec> is a file rather than a device, the file should be on a drive other than the one from which the directory listing is being produced. Sending a directory listing to a file on the same drive as that from which the directory listing is being produced will not, in general, work.

Example(s):

For example, issuing the command:

```
DIR D1: P:<RETURN>
```

will send to the printer a listing of all files on drive one. And the command:

```
DIR AB*.C??<RETURN>
```

will display on the screen a listing of all files on the diskette in the default drive which match the file specification "AB*.C??". Given that the diskette contains the following files, the table show which files will be listed and which won't:

```
ABX.CXX        listed
ABX.BXX        not listed
ACX.CXX        not listed
AB.CUR         listed
BB.CUR         not listed
ABCDEF.CNN     listed
ABCDEF.BNN     not listed
```

And, just:

```
DIR<RETURN>
```

will display on the screen a listing of all files on the diskette in the default drive.

4.2.5 How Do You ERAse Files From Command Level?

Command Syntax:

```
ERA[SE] [<drive-id>]<file-spec>
```

Command Token(s):

<drive-id>

This is an optional parameter which, if used, tells DOS XL to erase the files from the specified drive rather than the default drive. If <drive-id> is not included as part of the command, DOS XL will assume the default drive.

<file-spec>

This is a required parameter which tells DOS XL which file(s) are to be erased. Wildcard characters can be included as part of <file-

spec> (see "So What is a Diskette File Anyway?" for discussion of wildcard characters), thus permitting the erasure of certain groups of files.

Command Option(s):

None

Description:

The ERAse command permanently removes files from a disk. All files matching the <file-spec> on the specified drive (or the current default drive, if <drive-id> is omitted) will be erased from the disk. These files will no longer be shown when a DIRectory command is issued, nor will they be available for any type of file access. If DOS XL cannot find any files on the diskette which match <file-spec>, it will respond with "FILE NOT FOUND".

PROBLEM PREVENTOR: This command causes the irreversible deletion of files from the disk. It should be used with care. Use the PROtect command to guard files against accidental erasure.

Example(s):

Typing the command:

```
ERASE *.BAK<RETURN>
```

will erase all files with an extension of .BAK that are unprotected and that reside on the current default drive. The command:

```
ERA D2:DUP.SYS<RETURN>
```

will erase the file named DUP.SYS from disk in disk drive number two. All UNProtected files on the default drive can be erased with the single command:

```
ERA *.*<RETURN>
```

PROBLEM PREVENTOR: Use this command with caution!

4.2.6 How Do You LOAd a Binary File From Command Level?

Command Syntax:

LOA[D] [<drive-id>]<file-name>

Command Token(s):

<drive-id>
This is an optional parameter which, if used, tells DOS XL that the file to be LOAded should come from the specified drive rather than the default drive. If <drive-id> is not included as part of the command, DOS XL will assume the default drive.

<file-name>
This is a required parameter which tells DOS XL which file is to be LOAded.

Command Option(s):

None

Description:

The LOAd command allows the user to load binary load image files into user memory. The files must be compatible with the normal binary object files used by Atari's own DOS. Such files are produced by Atari's Assembler/Editor Cartridge as well as most upgraded products such as Optimized Systems' ACTION compiler and MAC/65 assembler.

PROBLEM PREVENTOR: LOAding binary image files is a DOS XL function required by very advanced programmers. Most DOS XL users have no use for this command, and therefore need not understand how to use it.

Example(s):

Typing:

```
LOA FASTGAME.COM<RETURN>
```

will cause DOS XL to LOAd the binary image file FASTGAME.COM into memory. DOS will not automatically transfer control to this file (start it executing) unless the file contains certain load information which causes DOS XL to transfer control. See "Using DOS XL With Assembly Language" for a full discussion.

4.2.7 How Do You PROtect Files From Command Level?

Command Syntax:

PRO[TECT] [<drive-id>]<file-spec>

Command Token(s):

<drive-id>

This is an optional parameter which, if used, tells DOS XL to protect the files on the specified drive rather than the default drive. If <drive-id> is not included as part of the command, DOS XL will assume the default drive.

<file-spec>

This is a required parameter which tells DOS XL which file(s) are to be protected. Wildcard characters can be included as part of <file-spec> (see "So What is a Diskette File Anyway?" for discussion of wildcard characters), thus permitting the protection of certain groups of files.

Command Option(s):

None

Description:

The PROtect command protects files on a disk from future ERAsure, RENaming, or any other alteration (including the writing of any data into the file). All files matching the <file-spec> on the specified drive (or the current default drive, if <drive-id> is omitted) will be protected. These files will then be shown on a DIRectory listing with asterisks/stars (*) in front of their

names. If DOS XL cannot find any files on the diskette which match <file-spec>, it will respond with "FILE NOT FOUND".

Example(s):

Typing the command:

```
PROTECT *.BAK<RETURN>
```

will protect all files with an extension of .BAK that reside on the current default drive. The command:

```
PRO D2:DUP.SYS<RETURN>
```

will protect the file named DUP.SYS on the disk in disk drive number two. All files on the default drive can be protected with the single command:

```
PRO *.*<RETURN>
```

4.2.8 How Do You RENname a File From Command Level?

Command Syntax:

REN[AME] [<drive-id>]<old-name> <new-name>

Command Token(s):

<drive-id>

This is an optional parameter which, if used, tells DOS XL to rename the file on the specified drive rather than the default drive. If <drive-id> is not included as part of the command, DOS XL will assume the default drive.

<old-name>

This is a required parameter which tells DOS XL which file is to be renamed.

<new-name>

This is a required parameter which tells DOS XL what the new name of

the file is to be.

Command Option(s):
None

Description:

The RENname command changes the name of a file on a disk from <old-name> to <new-name>. The renamed file will then be shown on a DIRectory listing with its new name. If DOS XL cannot find a file on the diskette which matches <old-name>, it will respond with "FILE NOT FOUND".

PROBLEM PREVENTOR: Although DOS XL will accept wild-card characters ("*" and "?") as part of <old-name>, RENname should not be used with wild-card characters. Using wild-card characters as part of the RENname command can permanently damage your diskette directory, preventing you from accessing your files.

PROBLEM PREVENTOR: When using both DOS XL and Atari DOS, it is possible to use the rename command to create two files with the same name. If this condition occurs, use the COPY command with the query (-Q) option to transfer the two files to separate disks where they may then be renamed back.

Example(s):

Typing the command:

```
RENAME MYFILE.NEW MYFILE.OLD<RETURN>
```

will change the name of the file "MYFILE.NEW" on the default drive to "MYFILE.OLD". The command:

```
REN D2:HELLO.TXT GOODBYE.TXT<RETURN>
```

will rename the file "HELLO.TXT" on the disk in disk drive number two to "GOODBYE.TXT".

4.2.9 How Do You RUN a Machine Language Program From Command Level?

Command Syntax:

RUN [<entry-address>]

Command Token(s):

<entry-address>

This is an optional parameter which, if present, tells DOS XL the memory address to which control is to be transferred in order to begin execution of the machine language program. The address must be supplied as a hexadecimal number. If <entry-address> is not supplied, the address contained in the DOS XL "RUNLOC" field is used instead. See "Using DOS XL With Assembly Language" for a discussion of "RUNLOC".

Command Option(s):

None

Description:

The RUN command allows the user to execute a binary load image file once it has been LOAded into user memory. The correct <entry-address> or contents of "RUNLOC" (see "Using DOS XL With Assembly Language" for a discussion of "RUNLOC") are very dependent upon the particular machine language to be executed using the RUN command.

PROBLEM PREVENTOR: RUNning binary image files is a DOS XL function required by very advanced programmers. Most DOS XL users have no use for this command, and therefore need not understand how to use it. Since there is no way for DOS XL to check and see if the <entry-address> supplied is actually the start of a valid machine language program, your Atari computer will probably "die" (stop talking to you) if you supply an invalid <entry-address> (or the contents of "RUNLOC" are invalid). If this happens, you will need to turn your Atari computer off and then on again in order to "bring it back to life" (reincarnate?) it.

SPECIAL NOTE: Many standard programs designed to be executed under DOS XL, as well as most of the DOS XL extrinsic commands, will set "RUNLOC" to their

special re-execute (warm start) entry address. Thus, for example, if you return to the DOS XL Command Processor to perform only intrinsic commands, you may be able to re-start the last program or DOS XL extrinsic command by simply typing "RUN<RETURN>". At the time of writing this, Optimized Systems' BASIC A+ and MAC/65 (for example) both support this capability: simply type "RUN<RETURN>" from Command Processor to re-start these two programs.

Example(s):

Typing:

```
RUN 4EFD<RETURN>
```

will cause DOS XL to transfer control of your Atari computer to the binary image file already loaded into memory and who's entry (first instruction) address is (hopefully) at 4EFD ("JMP $4EFD"). Alternately:

```
RUN<RETURN>
```

will cause DOS XL to transfer control of your Atari computer to the binary image file already loaded into memory and who's entry address is (hopefully) contained in the DOS XL memory location "RUNLOC" ("JMP (RUNLOC)").

4.2.10 How Do You SAVe a Binary File From Command Level?

Command Syntax:

```
SAV[E] [<drive-id>]<file-name> <starting-address> <ending-address>
```

Command Token(s):

<drive-id>

This is an optional parameter which, if used, tells DOS XL that the file is to be SAVed to the specified drive rather than the default drive. If <drive-id> is not included as part of the command, DOS XL will assume the default drive.

<file-name>

This is a required parameter which tells DOS XL the name of the file

under which to SAVe the binary memory image.

<starting-address>
This is the memory address, in hexadecimal, of the first byte which DOS XL is to write to the file.

<ending-address>
This is the memory address, in hexadecimal, of the last byte which DOS XL is to write to the file.

Command Option(s):
None

Description:

The SAVe command allows the user to write portions of memory to disk files in standard binary file format. The two addresses define the portion of memory to be written to disk; the second address must be greater than or equal to the first. A file which has been 'SAVed' may later be returned to memory using the LOAd command.

PROBLEM PREVENTOR: SAVing binary image files is a DOS XL function required by very advanced programmers. Most DOS XL users have no use for this command, and therefore need not understand how to use it.

Example(s):

The command:

```
SAVE PAGE4000.IMG 4000 40FF<RETURN>
```

will save the 256 byte "page" of memory at $4000 to the disk file PAGE4000.IMG on the default drive.

4.2.11 How Do You TYPe a File To The Screen From Command Level?

Command Syntax:

TYP[E] [<drive-id>]<file-name> <list-spec>

Command Token(s):

<drive-id>

This is an optional parameter which, if used, tells DOS XL which drive contains the diskette on which the file to be TYPed is located. If <drive-id> is not included as part of the command, DOS XL will assume the default drive.

<file-name>

This is a required parameter which tells DOS XL the name of the file you wish TYPed.

<list-spec>

This is an optional parameter which, if used, tells DOS XL where to send the typed file listing instead of the screen. This parameter can be used to specify either a device (such as "P:" for the printer), or another disk file.

Command Option(s):

None

Description:

The TYPe command instructs DOS XL to copy a text file to the screen, another file, or device. If the optional <list-spec> is not specified, the text file your specify will be copied to the screen.

Example(s):

To view the commands in the "STARTUP.EXC" file included as part of your DOS XL System Master diskette from Indus Systems, issue the command:

TYP STARTUP.EXC<RETURN>

(This command assumes that your DOS XL System Master diskette is in the default drive.) If the optional <list-spec> is specified, the text file will

be copied to the <list-spec> device or file. For example, to copy the "STARTUP.EXC" file to the printer, issue the command:

```
TYP STARTUP.EXC P:<RETURN>
```

Another use of the TYPe command is to create a short text file. For instance, you can create a new "STARTUP.EXC" batch file via the following:

```
TYP E: STARTUP.EXC<RETURN>
```

This command tells DOS XL that you wish to copy text from the screen E:ditor to the disk file "STARTUP.EXC". See "How Do You Use Your Other Non-Disk Devices" for a discussion of the screen E:ditor ("E:" device). When the screen E:ditor is the source "file", as in this example, you are actually the source "file" since input from the screen E:ditor comes from the Atari computer's keyboard. This means you will require a way to tell DOS XL when you are finished typing the "source file". To do this, you hold down the key labelled "CONTROL" or "CTRL", and then type the "3" key. This will create a signal to DOS XL which tells it that you have finished typing the "source file". When you use the screen E:ditor as the "source file", you cannot use the cursor movement keys to edit any line on except the line you are currently typing. It is this restriction which makes this command a little difficult to use for creating anything but short text files.

Finally, the TYPe command may also be used to copy TEXT files from one disk file to another by using disk file names for both <file-name> and <list-spec>. For example:

```
TYP STARTUP.EXC UPSTART.EXC<RETURN>
```

will copy the "STARTUP.EXC" file to the "UPSTART.EXC" file.

4.2.12 How Do You UNProtect Files From Command Level?

Command Syntax:

```
UNP[ROTECT] [<drive-id>]<file-spec>
```

Command Token(s):

<drive-id>
This is an optional parameter which, if used, tells DOS XL to unprotect the files on the specified drive rather than the default drive. If <drive-id> is not included as part of the command, DOS XL will assume the default drive.

<file-spec>
This is a required parameter which tells DOS XL which file(s) are to be unprotected. Wildcard characters can be included as part of <file-spec> (see "So What is a Diskette File Anyway?" for discussion of wildcard characters), thus permitting the unprotection of certain groups of files.

Command Option(s):
None

Description:

The UNProtect command unprotects files on a disk, permitting future ERAsure, RENaming, or any other alteration (including the writing of any data into the file). All files matching the <file-spec> on the specified drive (or the current default drive, if <drive-id> is omitted) will be unprotected. These files will no longer be shown on a DIRectory listing with asterisks/stars (*) in front of their names. If DOS XL cannot find any files on the diskette which match <file-spec>, it will respond with "FILE NOT FOUND".

Example(s):

Typing the command:

```
UNPROTECT *.BAK<RETURN>
```

will unprotect all files with an extension of .BAK that reside on the current default drive. The command:

```
UNP D2:DUP.SYS<RETURN>
```

will unprotect the file named DUP.SYS on the disk in disk drive number two.

All files on the default drive can be unprotected with the single command:

```
UNP *.*<RETURN>
```

4.3 DOS XL Intrinsic Batch Commands

Instrinsic batch commands differ from standard intrinsic commands since they are only functional when used inside a batch (.EXC) file. Like standard intrinsic commands, intrinsic batch commands differ from extrinsic commands by the fact that everything needed to process these commands is already contained within the DOS XL which was loaded into memory when you booted your system. DOS XL does not need to go back to your system diskette in order to get more information when you issue an intrinsic command from within a batch file, therefore you don't need to have your DOS XL diskette in the disk drive because your batch file contains one of these commands. However, you may still need your DOS XL diskette in the drive if that is where the batch file from which DOS XL is taking its commands is located.

The following is a summary of the intrinsic batch commands:

```
END
NOS[CREEN]
REM[ARK] [<any-text>]
SCR[EEN]
```

Note that all intrinsic batch commands may be abbreviated to their first three characters. As a matter of fact, DOS XL only looks at the first three characters while testing for an intrinsic batch command.

4.3.1 How Do You END a Batch File?

Command Syntax:
```
END
```

Command Token(s):
None

Command Option(s):
None

Description:

The END command causes DOS XL to stop reading commands from a batch file and to resume prompting the user for commmands. This command has no effect outside of a batch file.

Example(s):

Examples are provided under "How Do You Execute a Batch File From Command Level" and "Controlling DOS XL Without Being There", as well as with the examples for the other intrinsic batch commands.

4.3.2 How Do You Request NO-Screen Output From a Batch File?

Command Syntax:
NOS[CREEN]

Command Token(s):
None

Command Option(s):
None

Description:

Normally, all commands encountered during batch execution are displayed on the screen as if they were typed in by the user. The NOScreen command can be used to prevent the commands from displaying. All commands within the batch file will then no longer be displayed until the batch file is stopped for any reason or a SCReen command is encountered. This command is only effective from within a batch file.

Example(s):

Given the execution of a batch file containing the following commands:

```
DIR<RETURN>
NOS<RETURN>
REM THIS MESSAGE WILL NOT BE DISPLAYED
DIR<RETURN>
SCR<RETURN>
REM THIS MESSAGE WILL BE DISPLAYED
DIR<RETURN>
END<RETURN>
```

all three DIRectory listing will appear on the screen, but the second DIRectory command itself won't be displayed by DOS XL (the second DIRectory listing will appear to have been produced for no reason). The SCReen command will reverse the effect of the NOScreen command, thus permitting the third DIRectory command to be displayed. The NOScreen command will be displayed by DOS XL because it has yet to be executed and therefore the displaying of commands has yet to be suppressed. Similarly, the SCReen command will not be displayed since it has yet to be executed and therefore the displaying of commands has yet to be resumed.

4.3.3 How Do You Display REMarks From a Batch File?

Command Syntax:

REM[ARK] [<any-text>]

Command Token(s):

<any-token>

This can be any random text your little heart desires.

Command Option(s):

None

Description:

The REMark intrinsic batch command is used to place messages inside batch files which are displayed by DOS XL (provided NOScreen hasn't been executed) as the batch file execution progresses. DOS XL completely ignores this command except for displaying the command.

Example(s):

A batch file containing the following commands:

```
NOS<RETURN>
REM THIS MESSAGE WILL NOT BE DISPLAYED<RETURN>
SCR<RETURN>
REM THIS MESSAGE WILL BE DISPLAYED<RETURN>
END<RETURN>
```

will only display the second REMark command.

4.3.4 How Do You Request SCReen Output to Resume From a Batch File?

Command Syntax:
SCR[EEN]

Command Token(s):
None

Command Option(s):
None

Description:

Normally, all commands encountered during batch execution are displayed on the screen as if they were typed in by the user. The NOScreen command can be used

to prevent the commands from displaying. All commands within the batch file will then no longer be displayed until the batch file is stopped for any reason or a SCReen command is encountered. This command is only effective from within a batch file.

Example(s):

Given the execution of a batch file containing the following commands:

```
DIR<RETURN>
NOS<RETURN>
REM THIS MESSAGE WILL NOT BE DISPLAYED
DIR<RETURN>
SCR<RETURN>
REM THIS MESSAGE WILL BE DISPLAYED
DIR<RETURN>
END<RETURN>
```

all three DIRectory listings will appear on the screen, but the second DIRectory command itself won't be displayed by DOS XL (the second DIRectory listing will appear to have been produced for no reason). The SCReen command will reverse the effect of the NOScreen command, thus permitting the third DIRectory command to be displayed. The NOScreen command will be displayed by DOS XL because it has yet to be executed and therefore the displaying of commands has yet to be suppressed. Similarly, the SCReen command will not be displayed since it has yet to be executed and therefore the displaying of commands has yet to be resumed.

4.4 Extrinsic DOS XL Commands

The extrinsic commands are machine language programs which are run by DOS XL's Command Processor. Any standard Atari binary file containing the ".COM" extension may be used as a DOS XL extrinsic command. The DOS XL "COPY" command is one such extrinsic command. If you perform the DIRectory command to view the contents of your DOS XL System Master diskette, you will see a file named "COPY.COM". The program in the "COPY.COM" file is what is executed when the "COPY" command is typed.

Remember, extrinsic commands are not actually part of the DOS XL operating system. Whenever an extrinsic command is executed from DOS XL, the system must go looking on the diskette for a ".COM" file associated with the particular extrinsic command issued and load that file into your computer. For example, when you issue the extrinsic command "DUPDSK" the system will go looking on the diskette in drive one for a file called "DUPDSK.COM". If no ".COM" file having the same name as the command is on the diskette, the system will respond with a "FILE NOT FOUND" error. So remember: whenever you issue an extrinsic command to the system, its ".COM" file must be on the diskette for the command to execute properly.

Whenever you type any command to DOS XL, the first three characters of the command are compared to the intrinsic command list contained within the Command Processor. If the command is not in the intrinsic list, it is assumed to be extrinsic. Because the Command Processor first checks its own internal intrinsic command list, no extrinsic command program may start with three characters which match any of the intrinsic commands. For example, an extrinsic command program named "PROCESS3.COM" could not be executed by simply typing "PROCESS3", since DOS XL would view that as the intrinsic command "PROtect". There are two simple solutions to this problem should you encounter it:

1) Rename the extrinsic ".COM" file to a name which doesn't begin with an intrinsic command. Note: the ".COM" extension must still be used.

2) Execute the extrinsic command using a LOAd and then RUN instrinsic command sequence. For example, the extrinsic command "PROCESS3" could be executed by doing:

```
LOAD PROCESS3.COM<RETURN>
RUN<RETURN>
```

After the DOS XL Command Processor has searched its own internal list of intrinsic commands and found that the command you typed doesn't match any of those, it will do the following:

1) If you did not type a drive identifier ("D1:", "D2:", etc.) in front of your command, the Command Process will place the default drive identifier in front of your command.

2) Place the extension ".COM" after your command. Note: the Command Processor will place ".COM" after your command regardless of whether or not you specified a different extension, so it is not possible to override the ".COM" extension when issuing an extrinsic command.

3) Attempt to locate the ".COM" file on the specified drive. If it cannot locate the file, it will respond with a "FILE NOT FOUND" error.

4) Test the contents of the file to make sure it is proper Atari binary file format. See "How Do You LOAd a Binary File From Command Level?" for a discussion of the Atari binary file format.

5) Load the contents of the file (the program) into memory, and then execute the program.

NOTE: If the Command Processor has a problem performing any step in the above procedure, it will respond with an appropriate error message.

SPECIAL NOTE: The above procedure shows that you can prefix an extrinsic command with a drive identifier in order to cause the Command Processor to take the associated ".COM" file from a drive other than the default drive.

PROBLEM PREVENTOR: Never attempt to supply any extension (including ".COM") as part of any extrinsic command you type. As the above procedure shows, the Command Processor always appends the ".COM" extension to the extrinsic command you type regardless of when you have typed a different extension. This means that if you type the command "COPY.COM", the Command Processor will go looking for the program file "COPY.COM.COM" and, of course, not find it.

Most of the extrinsic commands you will use (such as "COPY") are supplied as part of your DOS XL System Master diskette. The number of possible extrinsic commands is not, however, limited to these few. If you have experience writing assembly/machine language programs for your Atari computer, you can write an extrinsic command to perform virtually any function. See "Using DOS XL With Assembly Language" for a discussion of how to write extrinsic commands.

SPECIAL NOTE: It is <u>not</u> possible to write extrinsic commands using Atari catridge BASIC. Extrinsic commands <u>must</u> be in a machine language form in

order for the Command Processor to execute them, and therefore either an assembler must be used (such as Atari's Editor/Assembler Cartridge), or a higher level language compiler (such as Optimized System's ACTION! Compiler).

Most of the extrinsic commands supplied on your DOS XL System Master diskette can be easily re-executed once they have terminated by simply issuing the RUN intrinsic command. Exceptions to this rule are the extrinsic commands "COPY", "SDCOPY", and "CONFIG". This feature may also work with other extrinsic commands you may have received from somewhere other than your DOS XL System Master diskette, it depends upon the particular extrinsic command.

PROBLEM PREVENTOR: You can only issue intrinsic commands between the time the extrinsic command terminated and you re-execute it using the RUN intrinsic command.

The following is a discussion of each extrinsic command supplied as a standard part of your DOS XL System Master diskette.

4.4.1 How Do You CLeaR a DiSK From Command Level?

Command Syntax:
CLRDSK

Command Token(s):
None

Command Option(s):
None

Description:

This utility is used to force your non-Atari disk drive to initialize a diskette just like the Atari 810 disk drive does. Hopefully any program that does not work with a diskette initialized in your non-Atari disk drive will work after you initialize the diskette using the CLRDSK extrinsic command.

NOTE: CLRDSK formats the diskette first, then writes zeroes to all sectors

execpt the directory, boot and VTOC sectors.

SPECIAL NOTE: Your Indus GT drive should not require the CLRDSK extrinsic command. The Indus GT initializes its single density diskette just like Atari's 810 drive when you use any of the other INITialization extrinsic commands supplied with DOS XL. The CLRDSK extrinsic command has been distributed as part of your Indus GT DOS XL System Master diskette in case you own some other manufacturer's drive which does not perform diskette initializations properly. (Not everybody's perfect.)

Example(s):

With the DOS XL System Master diskette in drive one, and the default drive set to one (Command Processor is prompting you with "D1:"), you would type:

```
CLRDSK<RETURN>
```

Once the Command Processor loads and executes CLRDSK.COM, CLRDSK will ask you:

```
Drive to clear?
```

You can then respond by typing "1" followed by the <RETURN> key. This will tell CLRDSK you wish to clear the diskette you are about to insert into drive one. CLRDSK will then prompt you with:

```
Insert disk and hit return
```

This prompt allows you to remove your DOS XL System Master diskette and insert the diskette you wish to clear.

PROBLEM PREVENTOR: Whichever diskette you insert at this time will completely lose any information it may have previously contained, so it is not a good idea to choose the diskette on which you've placed all of your relatives addresses and birthdates (unless, of course, you wish to be dis-owned by your entire family).

Once you've inserted an "okay to erase" diskette, press the <RETURN> key to tell CLRDSK to proceed. The clearing process takes about one minute and fifty seconds to perform when using an Indus GT. Other manufacturer's drives will

probably take longer. When CLRDSK has finished clearing your diskette, it will ask you:

Clear another disk?

to which you can either respond "Y" for yes followed by the <RETURN> key, or "N" for no followed by the <RETURN> key. If you respond with "N", CLRDSK will return you to the Command Processor. If you respond with "Y", CLRDSK will go back to the "Drive to clear?" question above.

4.4.2 How Do You CONFIGure Drive Densities From Command Level?

Command Syntax:

CONFIG [<setting>] [<setting>] [<setting>] [<setting>] [-N]

Command Token(s):

<setting>

These optional command parameters, if used, tells CONFIG how to configure one or more of the drives on your system. Each <setting> consists of two or three characters. The first character is always the number of the drive you wish to configure ("1"-"8"). The second one or two characters are used to specify the mode into which the drive is to be placed. "S" indicates single density, "D" indicates double density, and "DD" indicates double density and double sided. NOTE: Your Indus GT does not support the "DD" parameter, and the version of DOS XL supplied with your Indus GT does not support double sided drives. CONFIG does, however, still accept the "DD" parameter even though you should not use it.

Command Option(s):

-N

This command option, if used, tells CONFIG not to display the listing of how each drive attached to your system is configured.

Description:

If no parameters are given, this command simply reports the configuration of

all drives currently attached to your Atari computer.

If one or more parameters are given, they are presumed to be requests for CONFIG to tell the disk drives you specified (as the first character of each parameter) to configure themselves in the way you specified (as the second and third characters of each parameter). At the same time as CONFIG is telling your drives to change their operating modes/densities, it is also telling the DOS XL system itself that these changes are being performed. This is the very important difference between changing the operating densities of your Indus GT(s) using the CONFIG command rather than by using the front panel switches on your Indus GT(s). If you use the front panel switches on the Indus GT(s) to change their operating densities, the DOS XL system is never notified of the change, and therefore DOS XL and your Indus GT(s) get confused next time they try to talk to each other.

Normally, CONFIG will display a listing of all current configuration of all drives attached to your Atari. This listing will be suppressed if the "-N" option is used.

PROBLEM PREVENTOR: Do not try to re-CONFIGure the drive in which you have your DOS XL System Master diskette (usually drive one) unless you have a DOS XL System Master diskette recorded in the same density as that to which you are re-CONFIGuring the drive. If you do not have the second DOS XL System Master diskette available, then, on the drive changes densities, it will no longer be able to read the "CONFIG.COM" program file off your system master diskette and therefore you will be unable to re-CONFIGure your drive again. You will then be forced to re-boot your Atari computer.

PROBLEM PREVENTOR: At the time of writing this, Atari did not supply a disk drive which will respond to the CONFIG command. The Atari 810 drive is only single density, and therefore attempting to re-CONFIGure it would be useless anyway. The Atari 815 drive (a seldom seen beast) is only double density, and therefore attempting to re-CONFIGure it would also be useless. And last, but least, the Atari 1050 drive's enhanced density operating mode is not compatible with DOS XL (and most other drive manufacturer's double density modes), and therefore re-CONFIGuring the 1050 from single density (which is compatible with DOS XL) to double density (which isn't) would also be useless. Most other non-Atari disk drives support both single density and a DOS XL compatible double density, and therefore most of those drives will respond to a CONFIGure command. However, there may be limited cases in which a

manufacturer's double density drive will not respond to the DOS XL CONFIGure command. If you have one of these drives, you should know that the drive would not be considered "industry standard"; and therefore you should contact the drive's manufacturer (or your dealer) to see if your drive can be adjusted. NOTE: If you have a single density only drive, then it would be acceptable for it to not respond to the CONFIGure command since it does not support a double density operating mode. All Indus GT drives do support the CONFIGure command.

Example(s):

With the DOS XL System Master diskette in drive one, and the default drive set to one (Command Processor is prompting you with "D1:"), if you typed:

```
CONFIG 1D 2S<RETURN>
```

then once the Command Processor loads and executes CONFIG.COM, CONFIG would begin trying to "ask" each of the eight drives which DOS XL will support to answer back with their current operating density. If a particular drive number doesn't answer, CONFIG will assume that no drive is attached to your Atari with that given number. If the drive answers, but tells CONFIG that it doesn't understand CONFIG's request for operating density information, then CONFIG will assume that the drive is "unconfigurable" (such as Atari drives). If the drive does supply operating density information, CONFIG will assume that the drive is "configurable" and therefore CONFIG will accept a re-configuration parameter for that drive.

With the parameters listed in the example command above, if CONFIG found that drive one is configurable, it would then command drive one to change to double density operating mode. Likewise, if CONFIG found that drive two is configurable, it would then command drive two to change to single density operating mode. Parameters given for non-attached/non-existant or unconfigurable drives are ignored by CONFIG.

Since the option "-N" command option was not specified as part of the example command, CONFIG would then go ahead and display a table of the status of all eight of the possible drives. The last thing CONFIG will do before terminating back to the Command Processor is to tell DOS XL to re-initialize itself (this is not quite as dramatic as booting). This is necessary due to

the changes to the operating densities of some of the drives attached to your Atari computer.

4.4.3 How Do You COPY Files From Command Level?

Command Syntax:

COPY <source-spec> [<target-spec>] [-[F][Q][S][W]]

Command Token(s):

<source-spec>

This required parameter tells the COPY extrinsic command the name of the file (or group of files if the wild-card characters "*" and/or "?" is used) which is to be copied to the target file(s). See "So What is a Diskette File Anyway?" for a discussion of wild-card characters. This parameter can optionally include a drive identifier in front of the source file name. If the drive identifier is not used, the default drive is assumed.

<target-spec>

This is an optional parameter which, if used, tells the COPY extrinsic command the name of the file (or group of files if the wild-card characters "*" and/or "?" are used) which is to receive the source file(s). See "So What is a Diskette File Anyway?" for a discussion of wild-card characters. This parameter can optionally include a drive identifier in front of the target file name. If the drive identifier is not used, the default drive is assumed. If no target filespec if given, but a target drive identifier is given, then COPY will assume that the target names are to be the same as the source names. If you omit the entire <target-spec> parameter, COPY will assume you wish to copy the files between two different diskettes using only a single drive. The target names will be assumed to be the same as the source names in this case.

Command Option(s):

-F

Unless you specify this command option, COPY will not overwrite a target file which already exists.

-Q

If you specify this command option, COPY will display each file before it is copied, and then ask you if you desire the file to be copied.

-S

If you specify this command option, you will be informing COPY that the copy is to be performed between two different diskettes using a single drive. This option will cause COPY to prompt you to exchange your source and target diskettes at the appropriate times during the copying process.

-W

Unless you specify this command option, COPY will attempt to begin the copying process just as soon as it is executed by the Command Processor. If you specify this command option, COPY will tell you it is waiting for you to insert the proper diskettes before it begins the copying processes. This option is normally used when either the source or target drive is the one from which the Command Processor must load "COPY.COM", but the diskette containing "COPY.COM" is not one of the diskettes involved in the copying process.

Description:

The COPY extrinsic command copies one or more files without making any changes in the source file(s). If you provide a <target-spec>, all files matching the <source-spec> will be copied to the files indicated by the <target-spec>, which may be on the same or a different diskette. In you omit <target-spec> from the command line, those files indicated by the <source-spec> would be copied to files having the same name on the same drive but different diskettes. This enables the copying of files between diskettes using just a single drive system.

PROBLEM PREVENTOR: COPY cannot be used to copy between diskettes of two different densities using a single drive. The "SDCOPY" extrinsic command must be used for this purpose. COPY can be used to copy between diskettes of different densities on two different drives, or between diskettes of the same density (either single or double) on a single drive.

PROBLEM PREVENTOR: The COPY extrinsic command should never be used to copy the file "DOS.SYS" onto another diskette. DOS.SYS is a special file which is handled in a very special way by DOS XL (in order to remain fully compatible with Atari's own DOS). Because of this special handling, using COPY to copy DOS.SYS will result in the target DOS.SYS file containing two copies of the DOS.SYS file; and in many cases the target diskette will be unbootable. Only the INITialize extrinsic commands should be used to place a copy of DOS.SYS onto a diskette. After DOS.SYS has been placed onto a diskette using an INITialize extrinsic command, an "all files" copy can then be performed (by specifying the "*.*" file name). Since you will already have placed DOS.SYS on the target diskette, COPY will ask if it is "OK to overwrite?" the DOS.SYS file. Simply answer "N" for "no", and COPY will move on to copying the other files.

PROBLEM PREVENTOR: When using COPY with only a single drive, it will not bother to ask you to "Insert 'from' disk and hit RETURN" if you answered an "OK to overwrite?" question with "N" for "no". Therefore, always remember to insert your "from" disk before answering this question with "N". COPY will remember to ask you to "Insert 'from' disk and hit RETURN" after it is done copying the file if you answered "OK to overwrite?" with "Y" for "yes".

Example(s):

With the DOS XL System Master diskette in drive one, and the default drive set to one (Command Processor is prompting you with "D1:"), if you type:

```
COPY *.*
```

then once the Command Processor loads and executes COPY.COM, COPY will begin to copy all files on the current diskette in the default drive to another diskette to be placed in the same drive. COPY will prompt you with:

```
Insert 'to' disk and hit return
```

when it is time to swap diskettes. When you have swapped the diskettes, press the <RETURN> key to tell COPY that it may proceed. When COPY needs you to re-insert the first diskette again, it will prompt you with:

Insert 'from' disk and hit return

to which you should respond with the <RETURN> key once you've re-inserted the first diskette.

HELPFUL HINT: For this particular COPY function, the "DUPDSK" extrinsic command would generally be faster.

If you type the command:

```
COPY *.COM D3: -F
```

COPY will copy all files having an extension of ".COM" from the current diskette in the default drive to the current diskette in drive three (which, if drive three is the same as the default drive, will cause problems). If the file(s) already exist on the diskette in drive three, they will be erased and rewritten because of your specifying the "-F" command option.

If you type the command:

```
COPY D2:C*.* D1: -Q
```

COPY will ask you each time it finds a file starting with the letter "C" whether or not you wish the file to be copied from the diskette in drive two to the diskette in drive one. If you wish to tell COPY to copy the file, you should answer with "Y" for "yes" followed by the <RETURN> key. To tell COPY not to copy the file, you would answer with "N" for "no" followed by the <RETURN> key.

If you type the command:

```
COPY D1:TEST D2:NEWTEST
```

COPY will copy the file "TEST" on the diskette in drive one to the file "NEWTEST" on the diskette in drive two.

If you type the command:

```
COPY D1:TEST D1:NEWTEST -S
```

COPY will copy the file "TEST" on the diskette in drive one to the file "NEWTEST" on the diskette which you will place into drive one when requested by COPY.

4.4.4 How Do You DO Multiple Commands at Once From Command Level?

Command Syntax:

DO [<command>[;<command>[;<command>[;<command>[...]]]]]

Command Token(s):

<command>

This is an optional parameter which, if used, consists of various commands which you wish to issue in sequence. These <command>s are not restricted to just Command Processor intrinsic/extrinsic commands, see below.

Command Option(s):

None

Desription:

The DO command can be used in two ways. The first way allows you to issue several commands on one line. The semicolon (;) is used to separate commands which DO should pretend are on separate lines. The second way, in which you do not enter anything after the DO command, causes the DO extrinsic command to prompt you for each command line you wish to be "DO"ed.

PROBLEM PREVENTOR: Some extrinsic commands which inspect the Command Processor's command line themselves to find parameters for their own use cannot be used with the DO extrinsic command. One example of such an extrinsic command is the DOS XL "COPY" extrinsic command. "COPY" will not work correctly with DO because, on execution, COPY inspects the command line to find its source and target file specifications.

Example(s):

An example of how the DO command is not restricted to just Command Processor intrinsic/extrinsic commands would be:

```
DO CAR;RUN "D1:MENU"<RETURN>
```

This DO command would result in the issuing of, first, a Command Processor CARtridge instrinsic command; and, second, an Atari BASIC cartridge RUN command. The two commands would, first, start the Atari BASIC cartridge executing (if inserted into your computer); and, second, tell Atari BASIC to RUN the BASIC program "MENU" from drive one. This would be a very common command to place into a "STARTUP.EXC" file in order for your Atari computer to boot directly into running a BASIC program. See "What Does The STARTUP.EXC File Do?" for a further discussion.

As another example, the same sort of trick would work with Atari's Assembler/Editor Cartridge as well as Optimized System's MAC/65 Cartridge:

```
DO CAR;ENTER "D:PROGRAM.ASM";ASM ,#P:,#D:OBJECT<RETURN>
```

In this example, the Assembler/Editor Cartridge would first be executed; then the cartridge would be instructed to enter (load) the assembler source file "PROGRAM.ASM" from the default drive; and then the cartridge would be instructed to assemble the source file into machine language.

4.4.5 How Do You DUPlicate a DouBLe Density Diskette From Command Level?

Command Syntax:
DUPDBL

Command Token(s):
None

Command Option(s):
None

Description:

The DUPDBL extrinsic command will prompt you for the numbers of the drives containing the source and destination diskettes, and will ask whether it should format (initialize) the destination diskette prior to duplicating. The entire source diskette will then be copied to the destination diskette in a manner somewhat faster than the COPY extrinsic command would provide. The two diskettes, however, must be double density DOS XL diskettes originally formatted/initialized while using this version of DOS XL. If you specify that the destination drive is the same as the source drive, the program will prompt you to swap the diskettes when needed during the duplication process.

See "How Do You Copy Between Densities With Only One Indus GT?" or "How Do You Copy Between Densities Using Two Drives?" for a discussion of mixed density diskette duplication.

PROBLEM PREVENTOR: Do not attempt to use DUPDBL to duplicate single density diskettes. This can have disastrous results. The DUPDSK extrinsic command is designed for duplicating single density diskettes.

PROBLEM PREVENTOR: The DUPDBL extrinsic command will completely replace any information contained on whatever destination diskette you use with the information contained on the source diskette. If you use as a destination diskette a diskette which contains your secret formula for winning at the horse race track, you can forget about coming out ahead there for a while.

PROBLEM PREVENTOR: When DUPDBL is finished duplicating a diskette, it will ask you if you wish it to "Copy same disk again (Y or N) ?". You cannot safely duplicate a different source diskette by answering "Y" for "yes" to this question. In many cases, DUPDBL will still be holding some information in memory which came from the first source diskette, and if you tell DUPDBL to "Copy same disk again", it writes this information from the first source diskette to the new destination diskette; regardless of whether or not you changed source diskettes.

Example(s):

With the DOS XL System Master diskette in drive one, and the default drive set to one (Command Processor is prompting you with "D1:"), if you type:

```
DUPDBL<RETURN>
```

then once the Command Processor loads and executes DUPDBL.COM, DUPDBL will request:

Source Disk Drive (1,2,3,4):

to which you should respond with the number of the drive which (will) contain(s) your source diskette followed by the <RETURN> key. DUPDBL will then request:

Destination Disk Drive (1,2,3,4):

to which you should respond with the number of the drive which (will) contain(s) your destination diskette followed by the <RETURN> key. DUPDBL will then request:

Format Destination Disk (Y or N):

to which you should respond with "Y" for "yes" followed by the <RETURN> key if your destination diskette is brand new and has yet to be initialized, or "N" for "no" followed by the <RETURN> key if you are using an old diskette which was previously initialized.

At this point, DUPDBL is ready to begin; so it will prompt you with:

Put Source Disk In Drive #

where "#" is the drive number you specified as containing the source diskette. If you specified a different drive as containing the destination diskette, then DUPDBL will, at the same time, prompt you with:

Put Destination Disk In Drive #

where "#" is the drive number you specified as containing the destination diskette. The last thing DUPDBL will prompt you with is:

When Ready, Hit **RETURN**

You should place the indicated diskette(s) into the indicated drive(s) and then press the <RETURN> key to tell DUPDBL it can proceed. If you specified

two different drives as source and destination drives, then DUPDBL will be able to complete the entire duplication process without any further assistance from you. If you specified the same drive for both source and destination, DUPDBL will prompt you with:

Put Source Disk In Drive #

when it needs to read the source diskette, and:

Put Destination Disk In Drive #

when it needs to write the destination diskette. After either of these prompts, DUPDBL will prompt you with:

When Ready, Hit **RETURN**

to notify you that you need to press the <RETURN> key in order for DUPDBL to proceed. When DUPDBL is finished duplicating your diskettes, it will ask you:

Copy same disk again (Y or N) ?

If you wish to make another duplicate of the source diskette, you should repond with "Y" for "yes" followed by the <RETURN> key. If you do not wish to make another duplicate, you should respond with "N" for "no" followed by the <RETURN> key. (See Problem Preventor note above.) If you answer "yes", DUPDBL will prompt you for the destination disk. DUPDBL does not go back and re-ask you the drive number questions again. If you answer "no", DUPDBL will terminate and leave you back with the Command Processor.

4.4.6 How Do You DUPlicate a Single Density DiSKette From Command Level?

Command Syntax:
DUPDSK

Command Token(s):
None

Command Option(s):
None

Description:

The DUPDSK extrinsic command will prompt the user for the drive numbers containing the source and destination diskettes, and will ask whether to format (initialize) the destination diskette. The entire source diskette will then be copied to the destination diskette in a manner somewhat faster than the COPY extrinsic command would provide. The two disks, however, must be of the same size and type. If the destination drive is the same as the source drive, the program will prompt the user to swap diskettes during the duplication process.

See "How Do You Copy Between Densities With Only One Indus GT?" or "How Do You Copy Between Densities Using Two Drives?" for a discussion of mixed density diskette duplication.

PROBLEM PREVENTOR: Do not attempt to use DUPDSK to duplicate double density diskettes. This can have disastrous results. The DUPDBL extrinsic command is designed for duplicating double density diskettes.

PROBLEM PREVENTOR: The DUPDSK extrinsic command will completely replace any information contained on whatever destination diskette you use with the information contained on the source diskette. If you use as a destination diskette a diskette which contains your automobile's milage per tank-full for the last five years, you can forget about computing your average miles per gallon for a while.

PROBLEM PREVENTOR: When DUPDSK is finished duplicating a diskette, it will ask you if you wish it to "Copy same disk again (Y or N) ?". You cannot safely duplicate a different source diskette by answering "Y" for "yes" to this question. In many cases, DUPDSK will still be holding some information in memory which came from the first source diskette, and if you tell DUPDSK to "Copy same disk again", it writes this information from the first source diskette to the new destination diskette; regardless of whether or not you changed source diskettes.

Example(s):

With the DOS XL System Master diskette in drive one, and the default drive set

to one (Command Processor is prompting you with "D1:"), if you type:

DUPDSK<RETURN>

then once the Command Processor loads and executes DUPDSK.COM, DUPDSK will request:

Source Disk Drive (1,2,3,4):

to which you should respond with the number of the drive which (will) contain(s) your source diskette followed by the <RETURN> key. DUPDSK will then request:

Destination Disk Drive (1,2,3,4):

to which you should respond with the number of the drive which (will) contain(s) your destination diskette followed by the <RETURN> key. DUPDSK will then request:

Format Destination Disk (Y or N):

to which you should respond with "Y" for "yes" followed by the <RETURN> key if your destination diskette is brand new and has yet to be initialized, or "N" for "no" followed by the <RETURN> key if you are using an old diskette which was previously initialized.

At this point, DUPDSK is ready to begin; so it will prompt you with:

Put Source Disk In Drive #

where "#" is the drive number you specified as containing the source diskette. If you specified a different drive as containing the destination diskette, then DUPDSK will, at the same time, prompt you with:

Put Destination Disk In Drive #

where "#" is the drive number you specified as containing the destination diskette. The last thing DUPDSK will prompt you with is:

When Ready, Hit **RETURN**

You should place the indicated diskette(s) into the indicated drive(s) and then press the <RETURN> key to tell DUPDSK it can proceed. If you specified two different drives as source and destination drives, then DUPDSK will be able to complete the entire duplication process without any further assistance from you. If you specified the same drive for both source and destination, DUPDSK will prompt you with:

Put Source Disk In Drive #

when it needs to read the source diskette, and:

Put Destination Disk In Drive #

when it needs to write the destination diskette. After either of these prompts, DUPDSK will prompt you with:

When Ready, Hit **RETURN**

to notify you that you need to press the <RETURN> key in order for DUPDSK to proceed. When DUPDSK is finished duplicating your diskettes, it will ask you:

Copy same disk again (Y or N) ?

If you wish to make another duplicate of the source diskette, you should repond with "Y" for "yes" followed by the <RETURN> key. If you do not wish to make another duplicate, you should respond with "N" for "no" followed by the <RETURN> key. (See Problem Preventor note above.) If you answer "yes", DUPDSK will prompt you for the destination disk. DUPDSK does not go back and re-ask you the drive number questions again. If you answer "no", DUPDSK will terminate, leaving you with the Command Processor.

4.4.7 How Do You INITialize a diskette From Command Level?

Command Syntax:
INIT

Command Token(s):

None

Command Options(s):

None

Description:

The INIT extrinsic command allows you to initialize a diskette so that it may be read from or written to by other commands or programs. You must INITialize brand new diskettes before your Atari can read or write information to/from them.

PROBLEM PREVENTOR: INITializing a diskette which has existing programs or data on it is a sure way of sending those programs or data into the great computerized black hole, from which no program or data has ever returned. This is, of course, okay if you no longer need those programs or data.

PROBLEM PREVENTOR: The extrinsic command INIT is only used to INITialize diskettes in the same density as the current operating mode of the drive you select to perform the INITialization. If you only have a single drive and only a single density DOS XL System Master diskette, you must use the INITDBL extrinsic command in order to create a double density system master diskette. See "How Do You Make A Double Density DOS XL Diskette?" for a discussion of exactly that subject. If you have two or more drives, you can re-CONFIGure one of your secondary drives into double density mode, and then use this INIT extrinsic command.

PROBLEM PREVENTOR: You should always use diskettes equal in quality to the information which you are storing onto them. If you don't value the programs or data which you are storing onto your diskettes, then feel free to use poor quality diskettes. If you value the programs or data you are storing onto your diskettes, use only good quality diskettes. We at Indus feel that if you felt enough about quality to purchase our Indus GT, you should also be using equivalent quality diskettes. Your diskettes work as a team with your disk drives; and if one member of the team is no good, then the entire team will look bad.

PROBLEM PREVENTOR: In almost every case, you will need to INITialize a diskette which came from a non-Atari computer before you can use it on your

Atari. And this INITialization process will toss any programs or data into that same computerized black hole referenced above. Do not try to take a diskette from a non-Atari computer over to your Atari, then INITialize it, and then try to read any programs or data which was recorded by the other computer using your Atari. This is not the way to do this! Trying to read information from diskettes used on one type of computer using another type of computer is an extremely complex job. If you are trying to do this, it is possible that some software company might have a program which will assist you; but at the time of writing this, we at Indus did not know of the availability of such a program.

PROBLEM PREVENTOR: If you are using DOS XL with either an Optimized Systems' SuperCartridge or an Atari XL series computer, and the DOS XL System Master diskette from which you booted your system contains the file DOSXL.SYS, then do not use options two or three (see below) to write DOS.SYS (the DOS XL operating system file) to the disk. NOTE: Your original DOS XL System Master diskette from Indus does not contain a DOSXL.SYS file, we have left it up to you to create this file just to be safe (see "What Do The DOS.SYS, DOS.XL, DOS.SUP, and DOSXL.SYS files do?"). In order to use option two or three, you must re-boot your system using a diskette which does not contain a DOSXL.SYS file. You can do this by either renaming DOSXL.SYS to some other file name (such as just DOSXL) on your current system master diskette, re-boot your system, perform your INITialization, rename the file back to DOSXL.SYS, and finally re-boot your system again. Of course, an easier way to do this is to simply keep a system master diskette around which does not contain DOSXL.SYS on it. When you wish to INITialize, simply boot that diskette.

PROBLEM PREVENTOR: Options two and three of INIT (see below) only write DOS.SYS to the diskette. They do not write DOSXL.SYS. If you are using a DOSXL.SYS file on your system, you will need to COPY it from one of your existing system masters over to any new existing master you create.

The INIT extrinsic command, when executed, will present you with a menu of the following options:

1. FORMAT DISK ONLY
2. FORMAT DISK AND WRITE DOS.SYS
3. WRITE DOS.SYS ONLY
4. REFORMAT BOOT TRACKS ONLY
5. EXIT TO DOS XL

Option one can be used to create what is commonly referred to as a "data diskette". A "data diskette" is normally a diskette which does not contain the operating system (DOS XL in this case) and therefore cannot be booted. "Data diskette" is nothing more than a term someone invented to refer to such a diskette. There are no restrictions on what you can save onto a "data diskette". You can put program files, extrinsic command files, or any other types of files, as well as data files, onto a "data diskette". The advantage of creating a data diskette is that, without the DOS XL system files being on such a diskette, there is more room left for your own program or data files. It takes about 26 seconds for this option to be completed using an Indus GT, and possibly a little longer using a different diskette drive.

Option two creates a bootable system diskette. It starts by creating a "data diskette" just like in option one, but then finishes by adding the DOS XL system file "DOS.SYS" to the diskette. NOTE: Option two does not place any extrinsic command files, the MENU.COM (DOS XL menu) file, or the DOSXL.SYS file onto the diskette. These are all optional files as far as DOS XL is concerned, and therefore it leaves the COPYing of them onto the new diskette up to you. It takes about 35 seconds for this option to be completed using an Indus GT, and possibly a little longer using a different diskette drive.

Option three can be used to convert a "data diskette" into a "system diskette". NOTE: This option only writes DOS.SYS and a little bit of other information necessary to make the diskette bootable onto the diskette. This option assumes that you have previously INITialized the diskette using one of the other options. This option only writes the DOS.SYS file, just like option two. It takes about 10 seconds for this option to be completed using an Indus GT, and possibly a little longer using a different diskette drive.

Option four is used to reformat the boot or system tracks of a diskette that has been previously INITialized while Synchromesh was active. A full discussion of Synchromesh and its operation appears in Section 14 of this manual; a thorough understanding of the Synchromesh information should be had prior to using Option four of the INIT menu.

Option five is "the way out". It tells INIT that you are finished INITializing diskettes, and that you wish to return to DOS XL command level (or menu level if you came from the menu).

Example(s):

With the DOS XL System Master diskette in drive one, and the default drive set to one (Command Processor is prompting you with "D1:"), if you type:

```
INIT<RETURN>
```

then once the Command Processor loads and executes INIT.COM, INIT will request:

```
1. FORMAT DISK ONLY
2. FORMAT DISK AND WRITE DOS.SYS
3. WRITE DOS.SYS ONLY
4. REFORMAT BOOT TRACKS ONLY
5. EXIT TO DOS XL

ENTER FUNCTION NUMBER:
```

The normal selection if you wish to create a new bootable DOS XL diskette would be to choose option two (format disk and write DOS.SYS). REMEMBER: As explained in the Problem Preventor note above, this option cannot be used if you booted using a DOSXL.SYS file. To choose option two, type "2" followed by the <RETURN> key. After you have chosen any option (except the exit to DOS XL option), INIT will request:

```
ENTER DRIVE (1,2,3 OR 4):
```

INIT is asking you for the number of the drive which (will) contain(s) the diskette to be INITialized. Assuming you only have one drive, you would type "1" for drive one followed by the <RETURN> key. INIT will then confirm your entries shown below, then ask you:

```
FUNCTION 2; DRIVE 1

ARE YOU SURE (Y OR N):  (don't answer this question until you read below)
```

If you respond "Y" for "yes" to the yes/no question, INIT will immediately begin initializing the diskette in the drive you specified. If you respond "N" for "no", INIT will return you to the INITialize options menu. Assuming

everything is okay and you wish to proceed with initializing the diskette, make sure you have the diskette you wish to be INITialized in the drive you specified, and then respond with "Y" for "yes" followed by the <RETURN> key.

INIT will then proceed to perform whatever INITialization option you selected. Once INIT is finished, it will respond with:

```
TASK TERMINATED NORMALLY


HIT RETURN FOR NEXT FUNCTION
```

INIT will, of course, respond with an appropriate error message rather than "task terminated normally" if something went wrong during the INITialization process. To return to the INIT options menu, type the <RETURN> key. From the menu, you can either initialize another diskette or "exit to DOS XL". Assuming that you have just INITialized the only diskette you wish to INITialize, select option "4" from the menu and INIT will return you to DOS XL.

4.4.8 How Do You INITialize a DouBLe Density Disk From Command Level?

Command Syntax:
INITDBL

Command Token(s):
None

Command Options(s):
None

Description:

The INITDBL extrinsic command allows you to initialize a diskette in double density so that it may be read from or written to by other commands or programs. You must INITialize brand new diskettes before your Atari can read or write information to/from them.

PROBLEM PREVENTOR: Initializing (INIT/INITDBL) a diskette which has existing programs or data on it is a sure way of sending those programs or data into the great computerized black hole, from which no program or data has ever returned. This is, of course, okay if you no longer need those programs or data.

PROBLEM PREVENTOR: The extrinsic command INITDBL is only used to initialize diskettes in double density when the operating mode of the drive you select is single density. If you have more than one drive, it would probably be better if you re-CONFIGure one of your secondary drives into double density mode, and then use the INIT extrinsic command.

PROBLEM PREVENTOR: Please read all the Problem Preventor notes under "How Do You INITialize Diskette From Command Level?".

The only function INITDBL performs is essentially a double density version of option two when using the INIT extrinsic command: INITDBL reconfigures the disk drive you specify into double density operating mode, commands the drive to initialize your diskette, then INITDBL writes a copy of DOS.SYS (the DOS XL operating system file) onto the diskette so that the diskette is bootable, and finally reconfigures the drive back to single density. (Note that once INITDBL reconfigures the drive back to single density, the drive can no longer read the double density diskette which is in it.) Once INITDBL is finished, the diskette will be a bootable double density DOS XL system diskette, but it won't contain any of the extrinsic command files or the DOS XL menu program MENU.COM. (Note: "How Do You Make A Double Density DOS XL Diskette?" explains the creation process for a complete double density DOS XL System Master diskette.)

PROBLEM PREVENTOR: The drive you specify for use with INITDBL must start out in single density mode; and once INITDBL has finished its work, it will return the drive to single density mode. This means that the drive will not be able to read the double density diskette it just created unless you re-CONFIGure the drive to double density. See "How Do You CONFIGure Drive Densities From Command Level?" for a discussion of the CONFIG extrinsic command, as well as a Problem Preventor note relating to re-CONFIGuring your first (or only) drive.

Example(s):

With the DOS XL System Master diskette in drive one, and the default drive set to one (Command Processor is prompting you with "D1:"), if you type:

INITDBL<RETURN>

then once the Command Processor loads and executes INITDBL.COM, INITDBL will then ask:

DRIVE TO INITIALIZE?

Assuming you only have one drive, you would type "1" for drive one followed by the <RETURN> key. INITDBL will then respond with:

INSERT DISK AND HIT RETURN

This gives you a chance to change diskettes in the drive before initialization begins. Once you have the diskette to be initialized in the drive you specified, type the <RETURN> key and INITDBL will initialize the diskette. This process takes about 40 seconds using an Indus GT, and possibly longer when using other manufacturers' drives. When INITDBL is finished, it will automatically terminate back to the Command Processor.

4.4.9 How Do You Get to the DOS XL MENU From Command Level?

Command Syntax:
MENU

Command Token(s):
None

Command Option(s):
None

Description:

The MENU extrinsic command provides you with a simpler, although less powerful, way of commanding DOS XL to do your bidding. It was designed as a

way for the beginning user to "get acquainted" with the features of DOS XL before becoming swampped by the DOS's almost infinite capabilities. The MENU extrinsic command is all that need be issued to go from command level to the DOS XL menu.

The entire "The DOS XL Menu" section of this manual is devoted to a discussion of how you can use the DOS XL menu.

Example(s):

With the DOS XL System Master diskette in drive one, and the default drive set to one (Command Processor is prompting you with "D1:"), if you type:

```
MENU<RETURN>
```

then once the Command Processor loads and executes MENU.COM, MENU will display the standard DOS XL menu:

```
Files on Disk     Protect Files
To Cartridge      Unprotect Files
Copy Files        Rename File
Duplicate Disk    Save Binary
Erase Files       Load Binary
Initialize Disk   Go to Address
Xtended Command   Quit to DOS XL
```

The "The DOS XL Menu" section of this manual contains examples of using each one of the functions listed above.

4.4.10 How Do You Initialize Your Atari 850 RS-232 Ports?

Command Syntax:
RS232

Command Token(s):
None

Command Option(s):
None

Description:

If you do not own an Atari 850 Interface Module, then this command is of absolutely no use to you. The DOS XL RS232 extrinsic command is functionally equivalent to Atari DOS's AUTORUN.SYS file, which attaches the R: 850 RS-232 communications drivers to your operating system provided it finds an 850 Interface Module attached to your Atari Computer. When the RS232 extrinsic command is executed, it loads in a small piece of control software which remains in your system until you re-boot. This software handles the communications between your (or somebody else's) programs and the 850's RS-232 communication ports. This software is loaded into your computer's memory starting at the location pointed to by a system value called "LOMEM". Once the software is loaded, the value of "LOMEM" is adjusted to reflect the fact that this software is now resident inside your computer's memory.

PROBLEM PREVENTOR: If you do not own an 850 Interface Module then you need not understand this discussion at all. If you own an 850 Interface Module strictly for interfacing to a printer using the P: (printer) port and not the R: (RS-232) ports, then you need not understand this discussion. If you use software which requires the use of the RS-232 ports on your 850 Interface Module or you write strictly BASIC programs which use the RS-232 ports, then you should at least understand that part of the discussion which deals with executing the RS-232 extrinsic command. And finally, if you are an assembly language programmer, then you will also need to understand the adjustment of the "LOMEM" memory pointer. If you are an assembly language programmer, then you should also see "Using DOS XL With Assembly Language".

PROBLEM PREVENTOR: Due to a bug in the software contained with Atari's 850 Interface Module, hitting the <RESET> key will destroy the proper "LOMEM" value, effectively ignoring the space occupied by the RS232 communications software. See "What About Errors in the 850's RS-232 Driver?" for a discussion of a possible solution to this problem.

PROBLEM PREVENTOR: Atari's 850 Interface Module is sometimes too intelligent for its own good. In particular, you cannot generally re-execute the RS232 extrinsic command a second time without turning the 850 module off and back on

again. This is because the RS232 extrinsic commands requires a special response from the 850 module during initialization, and the 850 module will generally only provide that response once after having been turned on.

Example(s):

With the DOS XL System Master diskette in drive one, and the default drive set to one (Command Processor is prompting you with "D1:"), if you type:

```
RS232<RETURN>
```

then once the Command Processor loads and executes RS232.COM, RS232 will initialize your 850 Interface Module, set up the resident communications driver, and then terminate back to command level.

Examples relating to the use of the RS-232 ports on your Atari 850 Interface Module are contained in the manual which came with your 850 module.

4.4.11 How Do You Make a Single to Double Density COPY?

Command Syntax:

```
SDCOPY <source-spec> [<target-spec>] [-[F][Q][R][V]]
```

Command Token(s):

<source-spec>

This required parameter tells the SDCOPY extrinsic command the name of the file (or group of files if the wild-card characters "*" and/or "?" are used) which is to be copied to the target file(s). See "So What is a Diskette File Anyway?" for a discussion of wild-card characters. The parameter should not include a drive identifier since SDCOPY assumes you will be using drive one (see Discussion below).

<target-spec>

This is an optional parameter which, if used, tells the SDCOPY extrinsic command the name of the file (or group of files if the wild-card characters "*" and/or "?" are used) which is to receive

the source file(s). See "So What is a Diskette File Anyway?" for a discussion of wild-card characters. This parameter should not include a drive identifier in front of the target file name since SDCOPY assumes you will be using drive one (see Discussion below). If the drive identifier is not used, the default drive is assumed. If no target filespec if given, but a target drive identifier is given, then SDCOPY will assume that the target names are to be the same as the source names. If you omit the entire <target-spec> parameter, SDCOPY will assume you wish to copy the files between two different diskettes using only a single drive. The target names will be assumed to be the same as the source names in this case.

Command Option(s):

-F

Unless you specify this command option, SDCOPY will not overwrite a target file which already exists.

-Q

If you specify this command option, SDCOPY will display each file before it is copied, and then ask you if you desire the file to be copied.

-R

If you specify this command option, you will be informing SDCOPY that the copy is to be performed from a double density diskette to a single density diskette. This is reverse (-R) to SDCOPY's default direction of from a single density diskette to a double density diskette.

-V

If you specify this command option, SDCOPY will produce more verbose messages as the copying proceeds.

Description:

The SDCOPY extrinsic command copies one or more files, using only drive one, from a single density diskette to a double density diskette (or vice versa if you specify the -R command option), without making any changes in the source file(s). If you provide a <target-spec>, all files matching the <source-spec>

will be copied to the files indicated by the <target-spec>. If you omit <target-spec> from the command line, those files indicated by the <source-spec> will be copied to files having the same name on the target diskette.

PROBLEM PREVENTOR: SDCOPY cannot be used to copy between diskettes in two different drives. If you have more than one drive available, a far easier approach to copying between densities would be to CONFIGure one drive to single density, and the other drive to double density, and then simply use the standard COPY extrinsic command. SDCOPY is a specialized extrinsic command which, if you are a single drive user, provides you with a means of copying files between two different density diskettes.

PROBLEM PREVENTOR: The SDCOPY extrinsic command should never be used to copy the file "DOS.SYS" onto another diskette. DOS.SYS is a special file which is handled in a very special way by DOS XL (in order to remain fully compatible with Atari's own DOS). Because of this special handling, using SDCOPY to copy DOS.SYS will result in the target DOS.SYS file containing two copies of the DOS.SYS file; and in many cases the target diskette will be unbootable. Only one of the INITialize extrinsic commands should be used to place a copy of DOS.SYS onto a diskette. After DOS.SYS has been placed onto a diskette using an INITialize extrinsic command, an "all files" SDCOPY can then be performed (by specifying the "*.*" file name). Since you will already have placed DOS.SYS on the target diskette, SDCOPY will ask if it is "OK to overwrite?" the DOS.SYS file. Simply answer "N" for "no", and SDCOPY will move on to copying the other files.

PROBLEM PREVENTOR: SDCOPY will not bother to ask you to "Insert 'from' disk and hit RETURN" if you answered an "OK to overwrite?" question with "N" for "no". Therefore, always remember to insert your "from" disk before answering this question with "N". SDCOPY will remember to ask you to "Insert 'from' disk and hit RETURN" after it is finished copying the file if you answered "OK to overwrite?" with "Y" for "yes".

Example(s):

With the DOS XL System Master diskette in drive one, and the default drive set to one (Command Processor is prompting you with "D1:"), if you type:

```
SDCOPY *.*<RETURN>
```

then once the Command Processor loads and executes SDCOPY.COM, SDCOPY will begin to copy all files on the current single density diskette in the default drive to a double density diskette to be placed in the same drive. SDCOPY will prompt you with:

```
Insert 'to' disk and hit return
```

when it is time to swap diskettes. When you have swapped the diskettes, press the <RETURN> key to tell SDCOPY that it may proceed. When SDCOPY needs you to re-insert the first diskette again, it will prompt you with:

```
Insert 'from' disk and hit return
```

to which you should respond with the <RETURN> key once you've re-inserted the first diskette.

PROBLEM PREVENTOR: This is the only way to copy all the information from a single density diskette to a double density diskette when you only have a single drive. Because single density diskettes and double density diskettes have different sector sizes, none of the DUPlicate DiSK extrinsic commands can be used to convert information recorded on a single density diskette to double density format.

If you type the command:

```
SDCOPY *.COM -RF<RETURN>
```

SDCOPY will copy all files having an extension of ".COM" from the current double density diskette in drive one to a single density diskette to be placed in drive one when requested by SDCOPY. Note that the copy will be performed from a double density diskette to a single density diskette because you specified the "-R" for "reverse" option. If the file already exists on the single density diskette, it will be erased and rewritten because of your specifying the "-F" command option.

If you type the command:

```
SDCOPY C*.* -Q<RETURN>
```

SDCOPY will ask you each time it finds a file starting with the letter "C" whether or not you wish the file to be copied from the single density diskette to the double density diskette. If you wish to tell SDCOPY to copy the file, you should answer with "Y" for "yes" followed by the <RETURN> key. To tell COPY not to copy the file, you should answer with "N" for "no" followed by the <RETURN> key.

If you type the command:

```
SDCOPY TEST NEWTEST<RETURN>
```

SDCOPY will copy the file "TEST" from the single density diskette to the file "NEWTEST" on the double density diskette.

If you type the command:

```
COPY TEST NEWTEST -R
```

COPY will copy the file "TEST" on the double density diskette to the file "NEWTEST" on the single density diskette which you will place into drive one when requested by SDCOPY.

4.4.12 How Do You Cause DOS XL To VERIFY Or NOt VERIFY What It Writes?

Command Syntax:

```
VERIFY
  or
NOVERIFY
```

Command Token(s):

None

Command Option(s):

None

Description:

These utility programs allow DOS XL to write information to the disk drive in either of two modes. In the "VERIFY" mode, every sector written to the disk

is re-read and compared against the sector still in the computer's memory that was intended to have been written. If the sector does not pass this verifying check, an error 144 is returned to your computer and then to your program. In the "NOVERIFY" mode a re-read and compare after write is not performed.

It should be noted that the normal write process (without VERIFY turned on) produces extremely reliable results -- every sector written is re-read, and, although it is not compared with the sector in memory, it is checked for a correct CRC value. This CRC value is a complex mathematic operation that is performed on the data by the disk drive or computer prior to writing the particular sector. The chances are much less than one in a million that a sector could be written incorrectly and still match its originally calculated CRC value.

However, if you are running a program which is especially sensitive to data loss, or if you have a disk drive which is operating marginally, or if you are simply very cautious, you may wish to use DOS XL in the verify mode instead. To do so, simply type:

```
VERIFY
```

in response to the "D1:" prompt. Of course, to change your mind later, you may simply type:

```
NOVERIFY
```

instead.

Of course, to use either of these commands, the files VERIFY.COM and/or NOVERIFY.COM must be present on the disk. So use a copy of your master disk or copy these files to your working disk.

Section 5: THE DOS XL BOOT PROCESS

5.1 What Happens When DOS XL Is Booted?

The process of loading the DOS XL operating system into your Atari's memory is somewhat different than the process for loading other DOS's. Also, deleting or adding certain files to a bootable disk can affect what DOS XL will do while booting. In order for you to modify this process and thereby customize your system, this section describes the steps which are followed in the boot process.

5.2 What Do The DOS.SYS and DOSXL.SYS Files Do?

While most other DOS's reside only in the DOS.SYS file on a bootable disk, DOS XL actually occupies two separate files. The first file, DOS.SYS must be on any disk to make it bootable. At the beginning of the boot process, this file is loaded into memory. At that time, this DOS (it is actually a complete DOS in itself) checks to see if an Optimized Systems Software SuperCartridge is inserted. If not, the DOS begins to scan for the AUTORUN.SYS file (discussed later). If a SuperCartridge is inserted, the DOS tries to load the DOSXL.SYS file off of the disk. If this file is found it loads over the top of part of the DOS. already in memory and also into a portion of the SuperCartridge. This newly loaded code will then become the DOS of the machine. This DOS saves the user 5K of memory by occupying memory which is bank-switched with the SuperCartridge by taking advantage of special hardware within the cartridge. If you desire not to load this special DOS file, DOSXL.SYS, simply rename the file to a name other than DOSXL.SYS (perhaps SAVXL.SYS).

NOTE: The SuperCartridge is a product of Optimized Systems Software and is available from your local Atari dealer. For more information concerning the SuperCartridge, you should contact your dealer. The DOSXL.SYS version of DOS will not operate properly if Synchromesh is activated -- you can choose to take advantage of the extra storage that the DOSXL.SYS offers **OR** you can operate at Synchromesh speeds, but for the meantime, you cannot do both.

5.3 What Does The AUTORUN.SYS File Do?

Once the DOSXL.SYS file is either loaded or skipped, DOS XL searches the disk for a file called AUTORUN.SYS (note that there is no such file on the DOS XL master diskette which came with your Indus GT). If this file is found, it is loaded into memory just as if you had issued a "Load Binary" menu command.

The most common use Atari users find for the AUTORUN.SYS file is to load the RS-232 communications driver for their Atari 850 Interface Module into memory. Under DOS XL, this can be easily accomplished by simply renaming the file "RS232.COM" to "AUTORUN.SYS" on a copy of your DOS XL system diskette (never alter the original DOS XL System Master which came with your Indus GT). The RS232.COM file is discussed later in this manual.

5.4 What Does The STARTUP.EXC File Do?

If the file AUTORUN.SYS is not found, or if the program it contains returns to DOS in a special manner when it is completed (6502 RTS instruction), DOS XL will continue the boot process by searching for the file STARTUP.EXC. This file is a text file which contains commands for the DOS XL Command Processor to automatically perform. On your DOS XL System Master Diskette provided with your Indus GT there is a STARTUP.EXC file which contains REMark commands for just printing messages to the screen, and the command MENU, which causes the Command Processor to automatically load and start the DOS XL menu. (A different method of automatically loading the command menu is discussed later.)

PROBLEM PREVENTOR: Certain cartridge-based products, including Atari Writer from Atari, Inc., will not work properly if your boot disk contains a STARTUP.EXC file. If you are using a product such as Atari Writer, make a special boot disk as follows:

1) Duplicate your master disk onto a blank one.
2) Erase the file STARTUP.EXC on that disk.

You should now use this disk for booting DOS XL before running Atari Writer or other cartridge based products which won't function correctly when using a DOS XL diskette which contains a STARTUP.EXC file.

Section 5: THE DOS XL BOOT PROCESS

5.5 What Does The MENU.COM File Do?

If the STARTUP.EXC file is not found, the final step of the boot process is the loading of the DOS XL menu. The DOS will search the disk for the file MENU.COM. If that file is found, it will be loaded into memory and will be given control of your Atari. If the file MENU.COM is not found, the DOS XL Command Processor will remain in control. Regardless of whether control is passed to the menu program or left with the Command Processor, if there is a cartridge inserted, final control will be given to that cartridge. A full discussion of the command menu program is provided earlier in this manual.

PROBLEM PREVENTOR: When you use option 2 of the Initialize Disk function from the DOS XL Menu, the copy of DOS XL which will be written to your newly initialized diskette will expect to find the menu program (MENU.COM) when the new diskette is booted. But, the initialization utility does not place MENU.COM on your newly initialized diskette for you. In order to boot this new diskette, you must first use the "Copy Files" command menu item to copy MENU.COM from one of your DOS XL system diskettes onto your newly initialized diskette. If you forget to do this, the new DOS XL diskette will start to boot but will "die" (stop working) before it says anything on the screen.

Section 6: DOS XL AND THE 850 INTERFACE MODULE

6.1 What Is The 850 Interface Module?

The 850 Interface Module is a special communications interface produced by Atari. It gives your Atari computer the ability to communicate with other computer devices and peripherals which use standard parallel (usually printers) and RS-232 serial (many printers, plotters, modems, etc.) communications techniques. In order to use the RS-232 serial type of communications provided by Atari's 850 Interface Module, you must first load a special piece of software (called a "driver") into your Atari computer's memory. Atari DOS 2.0S automatically loads this RS-232 device driver into your Atari computer's memory whenever you boot DOS 2.0S. Under DOS XL, you can also cause the RS-232 driver to be automatically loaded when the DOS is

booted, but you also have other options.

If you don't happen to own an Atari 850 Interface Module, then you can save yourself some reading by skipping this section; you probably won't find it very useful.

6.2 How Do You Load The RS-232 Driver Under DOS XL?

When using Atari DOS 2.0S, the only way to load the RS-232 device driver (Rn:) contained in the 850 Interface Module is through the use of an AUTORUN.SYS file. This option is also available to you as a DOS XL user, as discussed earlier as part of the AUTORUN.SYS file. Another option is, however, available to you. After booting DOS XL, you can simply issue the following commands:

```
1) From the DOS XL menu:
          You type:      X
     (for eXtended command)
     and then, when prompted for a command,
          You type:      RS232<RETURN>

2) Or, from the DOS XL Command Processor:
          You type:      RS232<RETURN>
```

Either sequence of commands will cause the RS-232 device driver which is actually contained within the 850 itself to be loaded into your Atari computer. You can then use the four RS-232 serial ports on your Atari 850 Interface Module exactly as described in the Atari manual which came with your 850.

6.3 What About Errors In The 850's RS-232 Driver?

PROBLEM PREVENTOR: Unfortunately, the device driver which loads in from the 850 Interface Module is not perfect. The most serious flaw occurs when you push the [RESET] key after the RS-232 driver has been loaded into memory. Under certain circumstances, your Atari computer will "hang", freezing the keyboard, after pressing the [RESET] key. For this reason, many Atari

reference books recommend that you never press [RESET] after loading the RS-232 driver. Under DOS XL, however, there is a solution to this and other problems. On your master diskette which came with your Indus GT there is a file called "RS232FIX.COM". This file is almost identical to the "RS232.COM" file which is normally used to load the RS232 handler. The difference is that RS232FIX attempts to correct some of the known bugs in the driver when the driver is loaded.

You may ask, "Why not just include RS232FIX on the DOS XL master disk and leave off RS232?" Well, Atari has produced several versions of the 850 Interface Module. Indus and Optimized Systems Software has almost no way of knowing whether RS232FIX works correctly with all 850 versions so, rather than introducing new problems, both the original RS232 and the modified RS232FIX are included.

To test the "RS232FIX.COM" file with your 850 module, either:

1) Using the DOS XL menu:
   You type: X
   (for eXtended command)
   When prompted for a command,
   You type: RS232FIX<RETURN>

2) Or using the DOS XL command processor:
   You type: RS232FIX<RETURN>

If the RS-232 driver loaded in this way seems to work properly, you may use it exclusively for loading the RS-232 driver, ignoring the original RS232 command.

Section 7: SOME ADVANCED CAPABILITIES OF DOS XL

DOS XL is capable of performing several more advanced operations than those discussed in the previous chapters. Most of the advanced features are only accessable by using the DOS XL Command Processor (they are not directly contained as part of the command menu). Since a few of these advanced features may be required by beginning DOS XL users (depending upon exactly how many and which peripheral devices you own for your Atari), you may want to be or need to be familiar with these features. In particular, if you wish to transfer files from a single density diskette to a double density diskette (or vice versa) and/or if you wish to use more than two disk drives, you should read this section.

For most of the operations we will discuss in this section, you will be using the DOS XL Command Processor. As you probably recall from earlier discussions in this manual, you may leave the command menu and issue commands directly to the Command Processor by specifying the "Q" ("Quit to DOS XL") menu item. Once you've typed "Q" from the command menu, you will be greeted with the Command Processor's "D1:" prompt; as also discussed previously in this manual. (If you have not read these previous discussions, you should do so now.)

PROBLEM PREVENTOR: Always remember, when using the Command Processor's CONFIG, INIT, COPY, and SDCOPY commands, as described below, you must have a DOS XL system diskette in drive 1 when you type the command. If you do not, you will get a "FILE NOT FOUND" error message. This message will not be referring to any files you may have specified as part of your command, but rather to the fact that the utility program which the Command Processor required could not be found on the diskette.

7.1 How Do You Make A Double Density DOS XL Diskette?

This section assumes that you were shipped DOS XL in single density only. And since you have only a single density copy of DOS XL, we provide here two sets of step-by-step instructions on how to create a double density copy of DOS XL: one set of instructions for menu mode and one for command processor (advanced) mode. For both sets, the instructions assume that you have already booted a

single density DOS XL diskette; and that you are using your Indus GT as drive one. NOTE: It may be possible to use another manufacturer's double density drive as drive one, it all depends upon how the other manufacturer's drive was designed (just being double density isn't enough). To be safe, you should use your Indus GT.

PROBLEM PREVENTOR: Do not use your original master disk from Indus for the procedure you are about to follow. Be sure and use a copy of your master, instead. Since you may need to rename a file, you cannot have a write protect tab on the DOS XL diskette you use (the computer will need to write on the diskette). This can be very dangerous if you are using your only copy of DOS XL, hence the need for using only a copy of your master disk and never the original.

7.1.1 ...If I Want To Use The DOS XL Menu?

Step 1: Boot your copy of a single density DOS XL system diskette, then type "MENU" followed by the <RETURN> key. This will execute the DOS XL "MENU" program. NOTE: The MENU program will be automatically executed if you boot a duplicated copy of your DOS XL diskette from Indus. If your computer responds with BASIC's READY prompt, you must type DOS and then <RETURN> to reach the DOS and or DOS MENU.

Step 2: Select the "Files on Disk" menu item. DOS XL will then request "Filespec:". You should simply press the <RETURN> key. This step will give you a listing of all files on your DOS XL system diskette.

Step 3: Examine the list of files to see if the file "DOSXL.SYS" is listed. If it isn't, skip ahead to Step 7.

Step 4: Select the "Unprotect Files" menu item. DOS XL will then request "Filespec to unprotect:". You should respond with "DOSXL.SYS" followed by the <RETURN> key. This step will cause DOS XL to "unprotect" (allow you to rename or otherwise alter) the file "DOSXL.SYS". DOS XL has not actually altered the file, it has simply "unprotected" it.

Step 5: Select the "Rename File" menu item. DOS XL will then request "Old name:". You should respond with "DOSXL.SYS" followed by the <RETURN> key. DOS XL will next request "New name:". You should respond with "DOSXL"

followed by the <RETURN> key. And, just to be safe, DOS XL will ask "Are You Sure?". You should respond with "Y" for yes followed by the <RETURN> key. This step will cause DOS XL to change the name of the file "DOSXL.SYS" to just "DOSXL" to insure that you are not running an extended memory copy of DOS XL (see Problem Preventor note under "How Do You Initialize a Disk From the Menu?").

Step 6: Turn your computer's power off and then on again. This step will cause your computer to re-boot a standard (not extended memory) copy of DOS XL. If the copy of DOS XL you booted doesn't automatically execute the DOS XL menu for you, then skip ahead to Step 8.

Step 7: Select the "Quit to DOS XL" menu item. DOS XL will then respond with "D1:". This step will cause DOS XL to terminate the menu program and leave you in Command Processor (advanced) mode.

Step 8: Type "INITDBL" followed by the <RETURN> key. The INITDBL program will request "Drive to initialize?". You should respond with "1" for drive one and then the <RETURN> key. INITDBL will then request "Insert Disk and Hit Return". You should then remove your DOS XL system diskette and insert the diskette to be initialized. NOTE: This step will permanently erase any previous programs or data you may have on whatever diskette you insert (so please don't use the diskette containing your last seven year's tax records). Once you have inserted a diskette, press the <RETURN> key to tell INITDBL to begin. This process takes about forty (40) seconds using an Indus GT, and longer when using most other manufacturer's drives. When INITDBL is finished, "D1:" will be displayed. This step will initialize a diskette in double density.

Step 9: Remove the new double density diskette from the drive and re-insert your DOS XL system master diskette. Then type "SDCOPY *.* -Q" followed by the <RETURN> key. The first thing the program SDCOPY will do is request "Insert disk(s) to be copied and hit return when ready". Since you will be copying the DOS XL system master diskette which you already have in the drive, simply press the <RETURN> key to tell SDCOPY to begin. Each time SDCOPY is about to copy the next file on your DOS XL system master diskette, it will ask you if you wish that file copied. Since you wish all the files to be copied (with the exeception of DOS.SYS), you should respond with "Y" for yes followed by the <RETURN> key. For the file DOS.SYS, respond with "N" for no followed by the <RETURN> key. INITDBL

has already placed DOS.SYS onto your double density diskette, and if you tell SDCOPY to copy it, everything will get all messed up. DOS.SYS is the only file you should tell SDCOPY not to copy. SDCOPY should copy DOSXL.SYS, DOSXL.XL, and/or DOSXL.SUP (if any of them are present) along with any other files. Whenever you tell SDCOPY to go ahead and copy a file, it will sometimes request "Insert 'to' disk and hit return" or "Insert 'from' disk and hit return". The "to" disk is your double density diskette, and the "from" disk is your DOS XL system master. Whenever SDCOPY displays one of these messages, simply insert whichever diskette is being requested and then press the <RETURN> key. When the SDCOPY process has been completed, "D1:" will be displayed. This step copies all the files from your single density DOS XL system master diskette to your new double density DOS XL system master diskette.

Step 10: Type "MENU" followed by <RETURN> to get back to the DOS XL menu.

Step 11: If you did not perform Steps 4 through 6 above (your "Files on disk" display did not contain the file "DOSXL.SYS"), then you can skip ahead again to Step 14 at this point.

Step 12: Select the "Rename File" menu item. DOS XL will request "Old Name:". You should respond with "DOSXL" followed by the <RETURN> key. DOS XL will then request "New Name:". You should respond with "DOSXL.SYS" followed by the <RETURN> key. And last, DOS XL will ask "Are You Sure?". You should respond with "Y" for yes followed by the <RETURN> key. This step will rename the file DOSXL back to DOSXL.SYS.

Step 13: Select the "Protect Files" menu item. DOS XL will request "Filespec to Protect:". You should respond with "DOSXL.SYS" followed by the <RETURN> key. This step will cause DOS XL to re-"protect" the DOSXL.SYS file from any future alterations (until you ask DOS XL to "unprotect" it).

Step 14: Select the "Files on Disk" menu item. DOS XL will request "Filespec:". You should respond by typing just the <RETURN> key. DOS XL will then display the list of files on your single density DOS XL system master diskette. There are two things you should take note of on this display: the size (in number of sectors used) of the file "DOS.SYS" (this is the three digit number following the file name), and the number of "free sectors" which is displayed at the end of the list of file names. Remember both of these numbers for later.

Step 15: Turn your Atari computer off. Remove your single density DOS XL system master diskette from drive one. Insert your new double density DOS XL system master diskette into drive one. Turn your Atari computer back on again. This step will cause your Atari computer to re-boot DOS XL using the double density version.

Step 16: If, when you previously booted your single density DOS XL system master diskette, the menu program was automatically executed, then your new double density DOS XL system master diskette should also automatically execute the menu. If your double density DOS XL system master diskette doesn't automatically execute the DOS XL menu for you, you will need to type "MENU" followed by the <RETURN> key at this point (or type "DOS" and then <RETURN> if you were previously greeted by BASIC's READY prompt).

Step 17: If you did not perform Steps 4 through 6 above (your "Files on disk" display did not contain the file "DOSXL.SYS"), then you can skip ahead again to Step 22 at this point.

Step 18: Select the "Rename File" menu item. DOS XL will request "Old Name:". You should respond with "DOSXL" followed by the <RETURN> key. DOS XL will then request "New Name:". You should respond with "DOSXL.SYS" followed by the <RETURN> key. And last, DOS XL will ask "Are You Sure?". You should respond with "Y" for yes followed by the <RETURN> key. This step will rename the file DOSXL back to DOSXL.SYS on your double density DOS XL system master diskette.

Step 19: Select the "Protect Files" menu item. DOS XL will request "Filespec to Protect:". You should respond with "DOSXL.SYS" followed by the <RETURN> key. This step will cause DOS XL to re-"protect" the DOSXL.SYS file from any future alterations (until you ask DOS XL to "unprotect" it).

Step 20: If, at this point, you wish to re-boot your system using the extended memory version of DOS XL you were using before from your single density DOS XL system master diskette, turn your computer's power off and then on again. This step will cause your computer to re-boot the extended memory copy of DOS XL. If the copy of DOS XL you booted automatically executes the DOS XL menu for you, then skip ahead to Step 22. Again, if you are greeted by BASIC's READY prompt, typing "DOS" and then <RETURN> will turn control back over to DOS.

Section 7: SOME ADVANCED CAPABILITIES OF DOS XL

Step 21: Type "MENU" followed by the <RETURN> key. This step will execute the DOS XL menu.

Step 22: Remove your new double density diskette from the drive and place a write protect tab over the write protect notch!! Remember, this is your new double density DOS XL system master diskette. Master diskettes should only be used to create other "work" diskettes on which you can place your own programs or data. You should not use your master diskettes as work diskettes.

Step 23: Select the "Files on Disk" menu item. DOS XL will request "Filespec:". You should respond with just the <RETURN> key. DOS XL will then display all the files on your new double density DOS XL system master diskette. This listing will look just like the one from your single density diskette, with the exception of the number of sectors used by each file and the number of "free sectors".

Step 24: Compare the number of "free sectors" on the double density "Files on Disk" display with the number of "free sectors" you took note of from the single density "Files on Disk" display. You now have a lot more free sectors in which to store programs or data. (The miracles of modern science!)

Step 25: Compare the number of sectors used by the DOS.SYS files (the three digits following the file's name) with the number you took note of from the single density "Files on Disk" display. The reason why DOS.SYS uses half as many sectors on your double density diskette as it did on your single density diskette is because the sectors on your double density diskette are twice as large! This means all your files will use only half as many sectors on a double density diskette as they do on a single density diskette. So, not only does your double density DOS XL system master diskette have more free sectors available for use (since the files which are there are now taking up fewer sectors), but these free sectors can hold twice as much information. NOTE: A double density diskette has exactly as many sectors on it as a single density diskette, but double density sectors hold twice as much information as single density sectors. This means that only half as many double density sectors are required to hold the same amount of information, thus leaving more free sectors available for other information.

Step 26: Have fun with your new double density diskette and your Indus GT!!

7.1.2 ...If I Want To Use The Command Processor (Advanced) Mode?

Step 1: Boot your copy of a single density DOS XL system diskette. If the diskette you booted automatically executes the DOS XL menu for you, select the "Quit to DOS XL" menu item in order to get back to Command Processor (advanced) mode. If you are greeted with BASIC's READY prompt, type "DOS" and then <RETURN>.

Step 2: Type "DIR" followed by the <RETURN> key. This step will give you a listing of all files on your DOS XL system diskette.

Step 3: Examine the list of files to see if the file "DOSXL.SYS" is listed. If it isn't, skip ahead to Step 7.

Step 4: Type "UNP DOSXL.SYS" followed by the <RETURN> key. This step will cause DOS XL to "unprotect" (allow you to rename or otherwise alter) the file "DOSXL.SYS". DOS XL has not actually altered the file, it has simply "unprotected" it.

Step 5: Type "REN DOSXL.SYS DOSXL" followed by the <RETURN> key. This step will cause DOS XL to change the name of the file "DOSXL.SYS" to just "DOSXL" to insure that you are not running an extended memory copy of DOS XL (see Problem Preventor note under "How Do You Initialize a Disk From Command Level?").

Step 6: Turn your computer's power off and then on again. This step will cause your computer to re-boot a standard (not extended memory) copy of DOS XL. If the copy of DOS XL you booted doesn't automatically execute the DOS XL menu for you, then skip ahead to Step 8. If you are greeted with BASIC's READY prompt, typing "DOS" and then <RETURN> will cause you to leave BASIC and enter DOS.

Step 7: Select the "Quit to DOS XL" menu item. DOS XL will then respond with "D1:". This step will cause DOS XL to terminate the menu program and leave you in Command Processor (advanced) mode.

Step 8: Type "INITDBL" followed by the <RETURN> key. The INITDBL program will request "Drive to initialize?". You should respond with "1" for drive one and then the <RETURN> key. INITDBL will then request "Insert Disk and Hit Return". You should then remove your DOS XL system diskette and insert the diskette to be initialized. NOTE: This step will permanently erase any previous programs or data you may have on whatever diskette you insert (so please don't use the diskette containing your last seven year's tax records). Once you have inserted a diskette, press the <RETURN> key to tell INITDBL to begin. This process takes about forty (40) seconds using an Indus GT, and longer when using most other manufacturer's drives. When INITDBL is finished, "D1:" will be displayed. This step will initialize a diskette in double density.

Step 9: Remove the new double density diskette from the drive and re-insert your DOS XL system master diskette. Then type "SDCOPY *.* -Q" followed by the <RETURN> key. The first thing the program SDCOPY will do is request "Insert disk(s) to be copied and hit return when ready". Since you will be copying the DOS XL system master diskette which you already have in the drive, simply press the <RETURN> key to tell SDCOPY to begin. Each time SDCOPY is about to copy the next file on your DOS XL system master diskette, it will ask you if you wish that file copied. Since you wish all the files to be copied (with the exeception of DOS.SYS), you should respond with "Y" for yes followed by the <RETURN> key. For the file DOS.SYS, respond with "N" for no followed by the <RETURN> key. INITDBL has already placed DOS.SYS onto your double density diskette, and if you tell SDCOPY to copy it, everything will get all messed up. DOS.SYS is the only file you should tell SDCOPY not to copy. SDCOPY should copy DOSXL.SYS, DOSXL.XL, and/or DOSXL.SUP (if any of them are present) along with any other files. Whenever you tell SDCOPY to go ahead and copy a file, it will alternately request "Insert 'to' disk and hit return" or "Insert 'from' disk and hit return". The "to" disk is your double density diskette, and the "from" disk is your DOS XL system master. Whenever SDCOPY displays one of these messages, simply insert whichever diskette is being requested and then press the <RETURN> key. When the SDCOPY process has been completed, "D1:" will be displayed. NOTE: This step can be time consuming, but you shouldn't have to do it again some other time. This step copies all the files from your single density DOS XL system master diskette to your new double density DOS XL system master diskette.

Step 10: If you did not perform Steps 4 through 6 above (your "DIR" display did not contain the file "DOSXL.SYS"), then you can skip ahead again to Step 13 at this point.

Step 11: Type "REN DOSXL DOSXL.SYS" followed by the <RETURN> key. This step will rename the file DOSXL back to DOSXL.SYS.

Step 12: Type "PRO DOSXL.SYS" followed by the <RETURN> key. This step will cause DOS XL to re-"protect" the DOSXL.SYS file from any future alterations (until you ask DOS XL to "unprotect" it).

Step 13: Type "DIR" followed by the <RETURN> key. DOS XL will then display the list of files on your single density DOS XL system master diskette. There are two things you should take note of on this display: the size (in number of sectors used) of the file "DOS.SYS" (this is the three digit number following the file name), and the number of "free sectors" which is displayed at the end of the list of file names. Remember both of these numbers for later.

Step 14: Turn your Atari computer off. Remove your single density DOS XL system master diskette from drive one. Insert your new double density DOS XL system master diskette into drive one. Turn your Atari computer back on again. This step will cause your Atari computer to re-boot DOS XL using the double density version.

Step 15: If, when you previously booted your single density DOS XL system master diskette, the menu program wasn't automatically executed, then your new double density DOS XL system master diskette shouldn't automatically execute the menu. If your double density DOS XL system master diskette does automatically execute the DOS XL menu for you, you will need to select the "Quit to DOS XL" menu item. (If your screen shows READY, type "DOS" and then <RETURN>).

Step 16: If you did not perform Steps 4 through 6 above (your "DIR" display did not contain the file "DOSXL.SYS"), then you can skip ahead again to Step 20 at this point.

Step 17: Type "REN DOSXL DOSXL.SYS" followed by the <RETURN> key. This step will rename the file DOSXL back to DOSXL.SYS on your double density DOS XL system master diskette.

Step 18: Type "PRO DOSXL.SYS" followed by the <RETURN> key. This step will cause DOS XL to re-"protect" the DOSXL.SYS file from any future alterations (until you ask DOS XL to "unprotect" it).

Step 19: If, at this point, you wish to re-boot your system using the extended memory version of DOS XL you were using before from your single density DOS XL system master diskette, turn your computer's power off and then on again. This step will cause your computer to re-boot the extended memory copy of DOS XL. If the copy of DOS XL you booted automatically executes the DOS XL menu for you, then you will need to select the "Quit to DOS XL" menu item. (A READY message on your screen can be replaced by DOS prompts by typing "DOS" and then <RETURN>).

Step 20: Remove your new double density diskette from the drive and place a write protect tab over the write protect notch!! Remember, this is your new double density DOS XL system master diskette. Master diskettes should only be used to create other "work" diskettes on which you can place your own programs or data. You should not use your master diskettes as work diskettes.

Step 21: Type "DIR" followed by the <RETURN> key. DOS XL will then display all the files on your new double density DOS XL system master diskette. This listing will look just like the one from your single density diskette, with the exception of the number of sectors used by each file and the number of "free sectors".

Step 22: Compare the number of "free sectors" on the double density "DIR" display with the number of "free sectors" you took note of from the single density "DIR" display. You now have a lot more free sectors in which to store programs or data. (The miracles of modern science!)

Step 23: Compare the number of sectors used by the DOS.SYS files (the three digits following the file's name) with the number you took note of from the single density "DIR" display. The reason why DOS.SYS uses half as many sectors on your double density diskette as it did on your single density diskette is because the sectors on your double density diskette are twice as large! This means all your files will use only half as many sectors on a double density diskette as they do on a single density diskette. So, not only does your double density DOS XL system master

diskette have more free sectors available for use (since the files which are there are now taking up fewer sectors), but these free sectors can hold twice as much information. NOTE: A double density diskette has exactly as many sectors on it as a single density diskette, but double density sectors hold twice as much information as single density sectors. This means that only half as many double density sectors are required to hold the same amount of information, thus leaving more free sectors available for other information.

Step 24: Have fun with your new double density diskette and your Indus GT!!

7.2 How Do You Use Two Drives in Different Densities?

DOS XL version 2.35I is compatible with and capable of controlling any mixture of up to eight single density and/or double density (Indus GT) disk drives. If you have one or more drives only capable of single density (such as Atari's 810 or, when under DOS XL version 2.35I, Atari's 1050) as well as one or more Indus GT's capable of double density operation, we would suggest that you connect your double density Indus GT drive as drive 2 while creating your first double density DOS XL system diskette, and then as drive 1 for long-term usage. (See your Indus GT Owners Manual for switch settings, etc.). The long-term setup will allow you to boot DOS XL in either single or double density mode.

Although your Indus GT drive is capable of either single or double density operation, you can generally predict which density it will be in when power to your Atari computer is turned on. If it is drive 1, it will automatically acquire the density of the DOS XL diskette you are booting. If it is other than drive 1, your Indus GT will use the following rules to determine which density it is operating under, with rule 1 being first priority:

1) If there is a diskette in the drive, it will automatically change (if necessary) to operate in the density of that diskette when it receives its first command from DOS XL. (The only exception to this is if the first command is to format the diskette.)

2) If there is no diskette in the drive, but there has been a diskette read or written (accessed) by that drive since the drive was turned

on, the drive will remain in the density of the last diskette which was accessed.

3) If the drive has not accessed a diskette since it was turned on, it will default to using the density switch setting on the back of the drive. (See your Indus GT Owners Manual.)

PROBLEM PREVENTOR: Since rule 3 is last priority for your Indus GT in determining the density under which it will be operating, you can see that the switch on the back of the Indus GT is certainly not a "force the density" switch. The diskette currently in the drive, or the diskette which was last in the drive, has more influence over the drive's operating density than the back panel switch.

There are, of course, no switches to be set on single density only drives like the Atari 810 or, in the case of using DOS XL and even Atari DOS 2.0S, the Atari 1050 drive.

DOS XL as shipped is set up to handle one or two disk drives and up to three simultaneously open files in double density mode (i.e., Three BASIC "OPEN" statements without an intervening "CLOSE"). If you own three or more disk drives, or you require more files open at one time, you must change some of the settings contained within DOS XL. This is discussed later in this manual.

While booting, DOS XL automatically asks each drive the density under which the drive is operating. From then on, if you want to change a drive's density (by changing the diskette in the drive to one of a different density), you must use the CONFIG command from DOS XL's Command Processor. You should never use the front panel buttons to change the Indus GT's operating density since this will not notify DOS XL that a change has occurred.

CONFIG has several advanced options, which are discussed elsewhere in this manual. For our purposes, however, we need to learn three of its abilities before going on. Remember, the following commands must be performed using the Command Processor and not the menu, so you will have to use "Q" to get out of the menu first.

1. If you wish to find out what density DOS XL believes each drive is, following the Command Processor prompt:

D1:

You type: CONFIG<RETURN>

The table which is printed will tell you the density under which each disk drive is operating as well as other information regarding your drives. If a drive is not capable of double density operation, it is noted as "can't configure". Up to eight drives will be reported, and the system will even tell you which drives you don't have.

2. If you wish to change a drive from single to double density, following the Command Processor prompt:

D1:

You Type: CONFIG 1D<RETURN>

or you type: CONFIG 2D<RETURN>

or any other drive number which you wish to change.

In general, you specify the drive number you want to configure and then use a "D" to indicate double density. Any configurable drive may be changed this way. When CONFIG does its thing, it will also print the complete drive table to show you the change. If you first set your Indus GT to display the drive's type information (use the "DRIVE TYPE" button on the front of your Indus GT), you will be able to see the drive change from type "A" (single density, or "Atari 810 emulation mode") to type "b" (lower case "B" for double density, or "Atari 815 emulation mode").

PROBLEM PREVENTOR: If you use CONFIG to change the density of the drive in which you have your DOS XL system diskette, then the drive will no longer be able to read that diskette. Since CONFIG is a program which DOS XL must load off the diskette in order to change the drive's density back to its original density, this will cause you a problem unless you have already created a copy of your DOS XL diskette in the correct density. Essentially this means that you should wait to try this little CONFIG trick until you read through this entire discussion.

3. If you wish to change a drive from double to single density, following the Command Processor's prompt:

D1:

You Type: CONFIG 1S<RETURN>

or you type: CONFIG 2S<RETURN>

In general, you specify the drive number you want to configure and then use an "S" to indicate Single density. Any configurable drive may be changed this way. Just like when changing to double density, CONFIG will print a new drive table.

7.3 How Do You Initialize Different Density Diskettes?

As was noted in the previous discussion concerning the "Initialize Disk" menu item, that menu item should only be used to initialize a disk which is the same density as the booted master disk (or the current operating density of whichever drive is being used for the diskette initialization). Using the Command Processor, however, there are several other possibilities. And there is one exception.

1. If you have a single drive system, you may initialize a double density diskette even if you have booted a single density master (as is the case with the DOS XL System Master which came with your Indus GT). To do so, following the Command Processor prompt:

D1:

You Type: INITDBL<RETURN>

INITDBL is a Command Processor command which, when it has loaded from the DOS XL master disk, will simply ask you which drive you wish to use (presumably 1, in this case). It will then automatically configure the drive to double density, format the diskette, write DOS.SYS to the diskette, and then reconfigure the drive back to single density.

PROBLEM PREVENTOR: Do not use INITDBL when you have booted a double density master disk. Use menu option "I" instead.

2. If you have a multiple drive system, you may use CONFIG to configure drive 2 (or 3 or any other drive) to the density you desire. Then, following the Command Processor prompt:

D1:

You Type: INIT<RETURN>

and the INIT utility will load and run, presenting choices identical to those presented by the "I" menu option. Or, following the Command Processor prompt:

D1:

You Type: MENU<RETURN>

and the DOS XL menu will be reloaded and reactivated, and you may then use menu option "I". The actual rule is that you may initialize any diskette to the density of the drive it is placed in.

7.4 How Do You Copy Between Densities With Only One Indus GT?

This section applies only to those with one drive and that drive must be an Indus GT. If you have two or more drives, see the next section.

If you would like to copy one or more files from a single density diskette to a double density diskette (or vice versa), you must first have a diskette which has been formatted (initialized) to the proper density. If you do not have such a diskette, we suggest that you read the discussion above for assistance in initializing such a diskette.

Then, after having used the "Q" option of the menu, and following the Command Processor prompt:

D1:

you type:

SDCOPY D1:*.* D1:*.* -Q<RETURN>

or you type:

SDCOPY D1:*.* D1:*.* -QR<RETURN>

The first form will copy files from single to double density. The second form will copy files from double to single density.

SDCOPY will load in and then allow you to place your source ("from") diskette (in case its not the system diskette already in the drive) in the drive. SDCOPY will then read the directory of the source disk and give you a chance to say Yes or No about each file in the directory. If you answer Yes, the file will be copied to your destination ("to") disk. Since you have only a single drive, you will have to swap diskettes at least once for each file (long files may require two or three swaps). SDCOPY will tell you when it needs the diskettes to be swapped.

SUGGESTION: Remove any cartridges from your Atari if you are copying any large files. This may allow SDCOPY to copy these files in fewer swaps, since the cartridge space can then be used by SDCOPY. (This does not apply to Optimized Systems Software's SuperCartridges, which already automatically release their space to DOS XL.)

7.5 How Do You Copy Between Densities Using Two Drives?

If you own two or more drives, you may instead use the standard "Copy Files" command (option "C" of the menu) to transfer files between single and double density. First, however, you must ensure that your drives are appropriately configured.

Since you presumably have booted your double density master diskette on drive 1, we suggest that you use "CONFIG 2S" (discussed above) to place drive 2 in single density mode (unnecessary, of course, if drive 2 is an Atari 810 Disk

Drive, since it is always single density).

Remember, in order to use option "C" (for Copy) after using CONFIG, you must follow the Command Processor prompt:

D1:

with the command: MENU<RETURN>

This will command the Command Processor to reload the DOS XL menu and reactivate it. You may then choose option "C". When asked for "from" and "to" file names, be sure and specify "D1:" and "D2:", as appropriate. (If you make a mistake, Copy will probably not find the file names you are looking for, so no harm will be done. Just reverse the drive specifiers and try again.)

PROBLEM PREVENTOR: You may not use menu option "D" (Duplicate disk) to copy from a double density to single density diskette or vice versa! Strange and disastrous things will occur if you attempt to do so. Duplicated disks are literally duplicates, including the fact that the densities must be the same.

7.6 What Do You Do If You Have More Than Two Drives?

DOS XL, as it is shipped with your Indus GT, is set up to handle one or two disk drives and up to three simultaneously open files in double density mode (i.e., three BASIC "OPEN" statements without an intervening "CLOSE"). If you own three or more disk drives, or you require more files open at one time, you must change some of your DOS XL's default settings. DOS XL refers to the default setting associated with the number of drives you own as DRVBYT (you don't need to understand these names).

The default setting referred to as DRVBYT controls how many drives to which your DOS XL is willing to talk. DRVBYT is set to the value 3 (this doesn't mean three drives) on the DOS XL diskette which came with your Indus GT.

The following table gives the values to which DRVBYT will need to be changed in order to use more drives:

Value	Number of Drives

1	1 Only
3	1 or 2
7	1 to 3
15	1 to 4
31	1 to 5
63	1 to 6
127	1 to 7
255	1 to 8

Before you can change the DRVBYT value, you will need to create a new DOS XL system diskette (if you haven't already done so) which can be updated with the new value. Never alter the DOS XL System Master Diskette which came with your Indus GT.

Problem Preventor: The BASIC cartridge, like some other cartridges, have some conflicts when a STARTUP.EXC file exists on the copy of DOS that is booted into the computer's memory (this is true even if the BASIC cartridge is already built into your computer -- as with the 600XL and 800XL models). The following instructions may not work unless the copy of DOS which you boot prior to performing these steps is free of any STARTUP.EXC files.

In order to change the DRVBYT value, perform the following steps (this requires an Atari BASIC cartridge):

1) Turn you Atari computer's power off.

2) Insert your Atari BASIC cartridge into your computer (do not use Optimized Systems Software's BASIC XL). Boot your DOS XL master disk.

3) Insert the copy you made of your DOS XL System Master Diskette into drive 1.

4) Turn your computer on to boot DOS XL.

5) If your system disk stops booting at the command menu, terminate the menu program by specifying the "Quit to DOS XL" command.

6) Tell DOS XL to transfer control to the Atari BASIC cartridge by typing the Command Processor command "CAR<RETURN>".

7) After Atari BASIC gives you its "READY" prompt, type "POKE 1802,#<RETURN>";

where "#" in that typing sequence is one of the numbers from the table listed above.

8) Press your computer's [RESET] key.

9) After Atari BASIC responds with "READY", type "DOS<RETURN>". This will return control to DOS XL.

10) To get back to the DOS XL menu, following the "D1:" Command Processor prompt, type "MENU<RETURN>".

11) From the DOS XL menu, specify the "Initialize Disk" menu item.

12) When the INIT utility displays your four diskette initialization options, type "3<RETURN>" to specify the "WRITE DOS.SYS ONLY" option.

13) When the INIT utility asks you to "ENTER DRIVE (1,2,3 OR 4):", type "1<RETURN>" to specify that the modified DOS.SYS file is to be written to the diskette in drive 1.

14) The INIT utility will confirm your selections by displaying "FUNCTION 3; DRIVE 1" and "ARE YOU SURE (Y OR N):". If all this is correct, type "Y<RETURN>" to tell INIT to proceed. If you typed something wrong, type "N<RETURN>" and INIT will return you to the list of options.

15) Once you've typed "Y<RETURN>", INIT will write your modified copy of DOS XL to the DOS.SYS file on drive 1. To confirm that everything went correctly, turn your Atari computer off and then on again to reboot the new modified DOS XL. Once DOS XL is rebooted, you should be able to access as many drives as you selected from the above table.

7.7 How Do You Increase The Number Of Files You Can Have Open?

Using the copy of DOS XL which came with you Indus GT, you may open at the same time up to six single density files, or up to three double density files. If this setting is not enough for your needs, you will need to change one of DOS XL's default settings. DOS XL refers to this default setting as SABYTE (you don't need to understand this name). SABYTE is set to 6 on the copy of

DOS XL which came with your Indus GT. The following table lists other possible values to which you can change SABYTE:

Problem Preventor: The BASIC cartridge, like some other cartridges, have some conflicts when a STARTUP.EXC file exists on the copy of DOS that is booted into the computer's memory (this is true even if the BASIC cartridge is already built into your computer -- as with the 600XL and 800XL models). The following instructions may not work unless the copy of DOS which you boot prior to performing these steps is free of any STARTUP.EXC files.

SABYTE Value	Files Open in Single Density	Files Open in Double Density
2	2	1
4	4	2
6	6	3
8	--	4
10	--	5
12	--	6

Changing the value of SABYTE is almost identical to changing the value of DRVBYT as discussed above. The only difference is that for step 7 you would do:

7) After Atari BASIC gives you its "READY" prompt, type "POKE 1801,#<RETURN>"; where "#" in that typing sequence is one of the numbers from the table listed above. **NOTE:** The only difference between step 7 for changing DRVBYT and step 7 for changing SABYTE is that for DRVBYT the number after "POKE" is "1802", while the number after "POKE" when changing SABYTE is "1801".

Section 8: USING DOS XL WITH THE ATARI BASIC CARTRIDGE

This section discusses how to use your Indus GT disk drive and DOS XL from your Atari BASIC cartridge. This section is not intended as any kind of tutorial on how to use Atari BASIC or how to program in BASIC. Your Atari BASIC cartridge should have come with both a tutorial book which teaches you how to program in BASIC, and also a reference manual which lists each BASIC command and how it is used. This manual only includes the following discussion to explain how the same BASIC statements discussed in Atari's own manuals can be used to access files and information stored on your Indus GT diskette drive. The following discussion assumes you are already familiar with the information which Atari provides in their BASIC manuals.

But first, let's get the hardware set up right and things put in their proper places. Right off, you should have a copy of your DOS XL System Master Diskette which came with your Indus GT. Don't use your original diskette!! Take your copied master diskette and put it in drive 1, turn you Atari computer off, put your BASIC cartridge in the left cartridge slot, and turn the power on again.

Since you should be using a sucessful copy of the master disk by this point, we'll assume that the system booted properly (if it didn't, try everything again with a new blank diskette before calling us).

When DOS XL finishes booting and presents you with the DOS XL menu, you should select the "To Cartridge" menu item. When you do so, you should see BASIC's familiar "READY" prompt, and all is well.

Suppose, though, that you wanted to return to the DOS XL command menu. Why would you want to do that? Well, from the DOS XL menu you can list all files on a diskette, erase large collections of files, protect/unprotect large collections of files, and so on, without losing any BASIC program that you might have already typed in.

In fact, many of the menu commands (discussed earlier in this manual) may be used without harming even a line of what you may have typed in while in BASIC. To get back to your program just select "To Cartridge". You will be back in BASIC, and any program you might have written in is still there.

Section 8: USING DOS XL WITH THE ATARI BASIC CARTRIDGE

PROBLEM PREVENTOR: If you select any of the following menu items, you will lose any BASIC program you may have in memory:

Copy Files
Duplicate Disk
Initialize Disk
Xtended Command
Load Binary

If you need to perform any of these commands, save your BASIC program to disk prior to returning to the menu from BASIC.

So, from the DOS XL menu, "To Cartridge" will put you in BASIC (make sure the cartridge is there first). Nice and neat. But how do we get from BASIC back to the DOS XL menu? Simple: Following BASIC's "READY" prompt, type "DOS<RETURN>". This will return you to the DOS XL menu or command processor.

The following sections discuss the most common BASIC commands and statements which affect files on the disk. Please note that these commands should be issued while using the BASIC cartridge. That is, these commands should be typed only as part of a BASIC program or immediately after BASIC's "READY" prompt.

When the usage of the statement is being shown, the "{", "}", "[", and "]" characters are used to surround sections of the statement which are optional (not required). A "..." following a section of the statement infers that that section can be repeated zero or more times.

8.1 How Does BASIC's "CLOSE" Statement Work With DOS XL?

Statement: **CLOSE**

Purpose: This command writes any information which has been kept in memory and which is intended for the specified file to be written to the diskette, and then disassociates the file number (channel) and file which were associated by a previous OPEN statement.

Usage: CLOSE #fn

Arguments: "fn" -- file number (1-7)

Description:

After CLOSEing a file number, the user may no longer perform reads or writes (e.g, via PRINT, INPUT, etc.) on the file which had been associated with that channel.

PROBLEM PREVENTOR: A file OPENed for any form of output (modes 8, 9, or 12) should always be closed before the diskette containing it is removed or changed. The most common cause of crashed (no longer usable) Atari diskettes is failure to observe this rule.

HINT: Atari BASIC does not consider it an error to CLOSE a channel that is not OPEN, so it is often good practice to end a program segment by a line such as the following:

```
999 FOR I=1 TO 7 : CLOSE #I : NEXT I
```

NOTE: Both the END and RUN statements close all files (except file #0, which is used for keyboard and screen reads and writes), and can be used to advantage for this purpose when desired.

8.2 How Does BASIC's "ENTER" Statement Work With DOS XL?

Statement: **ENTER**

Purpose: This command is used to retrieve a BASIC program that has been LISTed to the disk.

Section 8: USING DOS XL WITH THE ATARI BASIC CARTRIDGE

Usage: ENTER filespec

Argument: "filespec" -- the name of the file you are going to ENTER.

Description:

The ENTER statement is used to retrieve a BASIC program that has been LISTed to the disk. As the program is being ENTERed into BASIC's user area, each line will be checked for proper syntax and converted into the internal (tokenized) form used by BASIC.

If a syntax error is encountered, the offending line will be listed with the suspected error location in inverse video.

NOTE: The line with the error will, nevertheless, be placed in program memory. In such a case, your program must be corrected before you can RUN it.

PROBLEM PREVENTOR: ENTER does not clear the user memory space. Therefore, if you wish to ENTER a new program, use NEW first. (Actually, this can be a handy feature when you wish to merge two programs together.)

Example:

```
You:      10 PRINT  "THIS IS PROGRAM 1"
          LIST "D:PROG1"
          10 PRINT  "THIS IS PROGRAM 2"
          LIST "D:PROG2"
          NEW
          ENTER "D:PROG1"
          LIST

BASIC:    10 PRINT  "THIS IS PROGRAM 1"

You:      NEW
          ENTER "D:PROG2"
          RUN

BASIC:    THIS IS PROGRAM 2
```

8.3 How Does BASIC's "GET" Statement Work With DOS XL?

Statement: **GET**

Purpose: This statement will retrieve a single byte of data from a specified previously OPENed disk file.

Usage: GET #fn,avar

Arguments: "fn" -- file number (1-7)
"avar" - any numeric variable

Description:

The GET statement is used to retrieve a single byte of data from a disk file that has been previously OPENed using the same file number which you specify on the GET statement.

NOTE: The data that you are GETting from the disk file should have been previously written to the specified file using the PUT statement.

Example program:

```
10 OPEN #1,8,0,"D:TEST" : REM CREATE A TEST FILE
20 FOR I = 0 TO 255 : PUT #1,I :NEXT I
30 CLOSE #1 :REM WE CREATED IT
40 OPEN #1,4,0,"D:TEST" : REM NOW CHECK IT OUT
50 FOR I = 0 TO 255 : GET #1,X : REM CHECK EACH
60 IF X <> I THEN PRINT "BAD DISK DATA",I,X
70 NEXT I
80 END : REM END CLOSES ALL FILES
```

8.4 How Does BASIC's "INPUT" Statement Work With DOS XL?

Statement: **INPUT**

Purpose: This command is used to request data from the specified file number (or keyboard).

Usage: INPUT {#fn,} var {,var...}

Arguments: "fn" -- file number (1-7)
"var" -- either numeric or string variable

Description:

When the INPUT statement is used without the "fn" option, data will be requested from the keyboard. You will notice a "?" appearing on the screen prompting you for the keyboard input. See your Atari BASIC Reference Manual for more details.

When the file number (#fn) argument is used, data will come in the form of ATASCII lines from the file that has been previously successfully OPENed using the same file number. Otherwise, the action of INPUT is virtually identical to the action when INPUTing data from the keyboard. That is, a string input is terminated by an ATASCII RETURN character and a numeric input by either the RETURN or a comma within a line.

NOTE: The INPUT statement cannot (generally) read a line that is longer than 127 characters in length. If you PRINT a line to the disk that you will later want to INPUT, it is best to limit the size of the PRINTed line to 127 characters or less.

Example program:

```
10 DIM LINE$(15)
20 OPEN #1,8,0,"D1:INPUT.SMP" : REM CREATE A FILE
30 FOR I = 1 TO 20
40 PRINT #1;"THIS IS LINE #";I : REM WRITE THE DATA
50 NEXT I
60 CLOSE #1 : REM CLOSE THE FILE YOU JUST CREATED
```

```
70 OPEN #1,4,0,"D1:INPUT.SMP" : OPEN FOR READ ONLY
80 FOR I = 1 TO 20
90 INPUT #1,LINE$ : REM GET THE FIRST LINE
100 IF LINE$(15) <> STR$(I) THEN GOTO 500
110 PRINT LINE$
120 NEXT I
130 CLOSE #1 : REM CLOSE THE FILE
140 PRINT "SUCESSFUL USE OF THE INPUT STMT"
150 STOP
500 REM WE GET HERE FROM LINE 100
510 PRINT "UNSUCCESSFUL USE OF INPUT"
520 END : REM ANOTHER WAY TO CLOSE THE FILE
```

Section 8: USING DOS XL WITH THE ATARI BASIC CARTRIDGE

8.5 How Does BASIC's "LIST" Statement Work With DOS XL?

Statement: **LIST**

Purpose: This command will LIST the program currently in memory to the screen (or to the file specified).

Usage: LIST [filespec]
or: LIST [filespec,] lineno1 [,lineno2]

Arguments: "filespec" -- the name of the file you are going to LIST to the disk.
"lineno1" -- beginning line number
"lineno2" -- ending line number

Description:

The LIST command is probably one of the most commonly used commands in BASIC. Most people know that the LIST command, when given all by itself, will LIST their program to the screen. Even when beginning and ending line numbers are given the results are predictable.

Now with DOS XL the LIST command can do even more. When used with a filespec, the LIST command will LIST your program to the disk instead of the screen. The contents of this file will contain text characters and can take up a large amount of disk space if you have a large program.

If you use the option where two line numbers are given, then only the lines from lineno1 to lineno2 (inclusive) will be LISTed to the filespec.

If you use the option where only one line number is given, then only that line will be LISTed to the filespec.

HINT: The ability to LIST a range of lines to the disk provides a convenient method of moving a subroutine (for example) to another program.

See also the ENTER statement discussion.

8.6 How Does BASIC's "LOAD" Statement Work With DOS XL?

Statement: **LOAD**

Purpose: This command will get a program that has been SAVEd to the disk and put it in BASIC's memory.

Usage: LOAD filespec

Arguments: "filespec" -- The name of the file you wish to LOAD.

Description:

LOAD is used in conjunction with the BASIC SAVE command. Only programs which have been previously SAVEd to disk may be LOADed. No syntax checking will be done as your program is being LOADed, because the program is already in internal format.

Generally, if you wish to keep a program on the disk, you SAVE it. Then, later, when you wish to look at it, modify it, or RUN it, you can LOAD it. BASIC does not remember the name that you use when you LOAD a program, so you can SAVE it again either under the same name (in which case the original version is replaced by the new version) or under another name.

Also, see the RUN command for an alternative method of LOADing a program which will simply be RUN and not modified.

Example:

```
You:      10 PRINT "THIS IS PROGRAM 1"
          SAVE "D:PROG1"
          10 PRINT "THIS IS PROGRAM 2"
          SAVE "D:PROG2"
          LOAD "D:PROG1"
          LIST

BASIC:    10 PRINT "THIS IS PROGRAM 1"

You:      RUN "D:PROG2"

BASIC:    THIS IS PROGRAM 2
```

8.7 How Does BASIC's "OPEN" Statement Work With DOS XL?

Statement: **OPEN**

Purpose: This command prepares a file for access and assigns it a file number.

Usage: OPEN #fn,aexp1,aexp2,filespec

Arguments: "fn" -- file number (1-7)
"aexp1" -- Read or Write (Input/Output) Mode:
4 = input
6 = directory access
8 = output
9 = append
12 = input/output

"aexp2" -- device dependent value (usually 0)
"filespec" -- a proper DOS XL file name

Description:

The OPEN statement allows a disk file (or any device, for that matter) to be linked to a file number (channel) for future reference in connection with file input/output instructions (e.g., PUT, GET, INPUT, PRINT, CLOSE).

Comments on arguments:

The "fn" argument allows for a number between 1 and 7. The number 0 is reserved for the screen and can not be used in Atari BASIC. After a file has been OPENed with a given "fn", all references to that file must be made using that same "fn".

The "aexp1" argument allows the user to OPEN a file for a specific "mode", according to the following table:

Mode 4: will OPEN the specified file for input only. Thus you can only retrieve data from the specified file.

Mode 6: allows you to access the directory on the disk.

Mode 8: is the opposite of mode 4. That is, data can only be stored to the specified file. See below for notes when using mode 8.

Mode 9: is used to add data to the specified file. The data that is added will begin at the current end of the specified file.

Mode 12: is used to access the specified file for input and output. Thus data can be stored and retrieved from the specified file.

PROBLEM PREVENTOR: After OPENing a file, the specified file number must be used to designate the file in other input/output statements. Two OPENed files cannot have the same file number, but it is possible to OPEN the same file with two different file numbers. Generally, such a double OPEN will have disastrous results.

NOTE: If a file is OPENed for output (aexp1=8) and the specified file does not exist then a file with the specified name will be created for you. If the file specified already exists, it will be destroyed and a new file with the specified name will be created for you.

PROBLEM PREVENTOR: A file OPENed for update (aexp1=12) can not be appended to under DOS XL or Atari DOS 2.0S. Only files opened in mode 9 will allow a file's size to be increased.

HINT: Mode 6 might, for example, be used from BASIC to find what files are on a disk and thereby allow a menu selection. The following program will allow a menu selection of all BASIC SAVEd programs on drive 1, providing that the program names do not have an extension (i.e., the programs should not have been SAVEd as "D:name.ext" but simply as "D:name").

Example program:

```
100 OPEN #1,6,0,"D:*" : DIM LN$(40)
110 FOR I = 1 TO 20 : INPUT #1, LN$
120 IF LN$(2,2)=" " THEN PRINT I,LN$(3,10) : NEXT I
130 CLOSE #1 : OPEN #1,6,0,"D:*"
140 PRINT : PRINT "WHAT PROGRAM TO RUN ";
150 INPUT J : IF J>=I THEN GOTO 140
160 FOR I = 1 TO J : INPUT #1,LN$ : NEXT I
```

```
170 CLOSE #1 : LN$(1,2) = "D:"
180 RUN LN$(1,10)
```

Try typing this in and then saying SAVE "D:MENU". Later, you can use the program by typing RUN "D:MENU".

PROBLEM PREVENTOR: As shipped by Indus, your DOS XL will allow only six single density files or three double density files to be OPEN at one time. See Section 7.7 (above) for "How Do You Increase The Number Of Files You Can Have Open?".

Section 8: USING DOS XL WITH THE ATARI BASIC CARTRIDGE

8.8 How Does BASIC's "PRINT" Statement Work With DOS XL?

Statement: **PRINT**

Purpose: This command puts the ASCII equivalents of the given expressions to the file specified on the screen.

Usage: PRINT [#fn {;}] exp [{,}exp...] {,}

Arguments: "fn" -- file number (1-7)
"exp" -- the expression can either be a string enclosed in double quotes, a string variable, or a numeric variable.

Description:

When a file number is used with the PRINT command, the specified variables are PRINTed to the disk file that has been previously OPENed using the same file number.

NOTE: Characters are PRINTed to a disk file in a manner identical to the way characters are PRINTed to the screen if the file number option is not used.

PROBLEM PREVENTOR: A "," after the #fn causes tabbing before the first character is PRINTed. A ";" does not cause the tabbing. Normally, the semicolon should be used.

See also INPUT.

8.9 How Does BASIC's "PUT" Statement Work With DOS XL?

Statement: **PUT**

Purpose: This statement is used to store a single byte of data to a specified file.

Usage: PUT #fn,avar

Arguments: "fn" -- file number (1-7)
"avar" -- an arithmetic variable

Description:

The PUT statement is used to output a single byte of data to a specified file. The file number used in the PUT statement must be one that has been previously used in the successful OPEN of a file.

PROBLEM PREVENTOR: Data that has been stored in a file using the PUT statement can usually only be retrieved using the GET statement.

See also GET.

Section 8: USING DOS XL WITH THE ATARI BASIC CARTRIDGE

8.10 How Does BASIC's "SAVE" Statement Work With DOS XL?

Statement: **SAVE**

Purpose: This command will store a BASIC program on disk in internal format (not ATASCII).

Usage: SAVE filespec

Arguments: "filespec" -- filename you wish to SAVE your program under (remember, a proper filespec includes the device specifier D: or D1:, etc. when the file is to be read from or written to a disk drive).

Description:

The SAVE command is used to SAVE your BASIC program in its internal format. This format is usually smaller than the text form of your program and will take up less room on your disk. All programs SAVEd to the disk must be reentered using the LOAD or RUN commands.

See descriptions of LOAD and RUN for more examples and further explanations.

8.11 How Does BASIC's "XIO" Statement Work With DOS XL?

Statement: **XIO**

Purpose: This is BASIC's catch-all Input/Output command. If BASIC doesn't provide a function to access a particular feature of a device or file, some form of XIO can probably be used to do so.

Usage: XIO subcommand,#fn,aux1,aux2,filespec

Arguments: "command" -- see descriptions below.
"fn" -- a file number. In contrast to most DOS XL read/write commands, XIO often requires that the file number be that of an un-OPENed channel. The XIO command dictates the usage here, so see descriptions below.
"aux1" and "aux2" -- generally zero. These values are passed to DOS XL unchanged (and thence to the device being accessed), so the individual device(s) may require other values. None of the examples given in this section use these values.
"filespec" -- a proper DOS XL file name.

Description:

Although, as noted, XIO can be used for several purposes, we will restrict our discussion here to those four XIO commands most useful to the Atari BASIC programmer.

The XIO commands to be discussed will each be treated as separate BASIC statements.

8.12 How Do You Rename A File From Atari BASIC?

XIO Command: **32 (Rename)**

Purpose: May be used to rename disk files.

Usage: XIO 32,#fn,0,0,filespec

Arguments: "fn" -- the file number of an un-OPENed channel.
"filespec" -- a proper DOS XL file name followed by, in the same BASIC string, a comma and a second file name. The second file name may not include a disk drive specifier.

Description:

It is suggested that "fn", the file number, be 7, since that channel is normally reserved for system read/write functions (which this certainly is). The only thing strange about this XIO command is the form of the filespec. Some examples follow:

```
XIO 32,#7,0,0,"D:TEST.SAV,OLDTEST.SAV"

DIM FL$(100)
INPUT FL$
FL$(LEN(FL$)+1) = ",BACKUP"
XIO 32,#7,0,0,FL$
```

Again, note that the second file name in both examples is not preceded by a disk drive specifier.

8.13 How Do You Erase A File From Atari BASIC?

XIO Command: **33 (Erase/Kill/Delete)**

Purpose: May be used to permanently erase disk files.

Usage: XIO 33,#fn,0,0,filespec

Arguments: "fn" -- the file number of an un-OPENed channel.
"filespec" -- a proper DOS XL file name, with "wild cards" accepted and processed.

Description:

If the file specified exists on the disk drive specified, and if the file is not PROTECTED (see next XIO command), the specified file will be permanently erased (deleted, killed, zapped) from the disk.

USE THIS XIO SUBCOMMAND WITH CAUTION: specifying a "wild card" (a file name including an asterisk or question mark) will erase all files which match the given name.

Examples:

```
XIO 33,#7,0,0,"D2:OLDPROG.SAV"
```
will erase the single file with the name OLDPROG.SAV from the diskette in drive 2.

```
XIO 33,#5,0,0,"D:*.BAK"
```
will erase all files having a filename extension of ".BAK" from the diskette in drive 1.

8.14 How Do You Protect Files From Atari BASIC?

XIO Command: **35 (Protect/Lock)**

Purpose: May be used to protect disk files from accidental erasure and modification.

Usage: XIO 35,#fn,0,0,filespec

Arguments: "fn" -- the file number of an un-OPENed channel.
"filespec" -- a proper DOS XL file name, with "wild cards" accepted and processed.

Description:

All files on the specified drive which have names which match the specified file will be "PROTECTED" by usage of this XIO command. Protection in the DOS XL environment simply consists of setting a flag in the diskette's file directory which tells the DOS to disallow either modification (i.e., OPENs in modes 8, 9, 12, etc.) or erasure of the file. Any DOS XL DIRectory listing will show protected files by means of an asterisk in the first column of the displayed lines (unprotected files have simply a space in that position).

Examples:

```
XIO 35,#7,0,0,"D:*.*"
```
will protect ALL files on drive 1.

```
XIO 35,#1,0,0,"D4:DOS.SYS"
```
will protect only the file named "DOS.SYS" on the diskette in drive 4.

8.15 How Do You Unprotect Files From Atari BASIC?

XIO Command: **36 (Unprotect/Unlock)**

Purpose: May be used to unprotect disk files to allow subsequent erasure and modification.

Usage: XIO 36,#fn,0,0,filespec

Arguments: "fn" -- the file number of an un-OPENed channel.
"filespec" -- a proper DOS XL file name, with "wild cards" accepted and processed.

Description:

All files on the specified drive which have names which match the specified file will be "UNPROTECTED" by usage of this XIO command. Protection in the DOS XL environment simply consists of setting a flag in the diskette's file directory which tells the DOS to disallow either modification (i.e., OPENs in modes 8, 9, 12, etc.) or erasure of the file. Any DOS XL DIRectory listing will show UNprotected files by means of a space in the first column of the displayed lines (protected files have an asterisk in that position).

Examples:

XIO 36,#7,0,0,"D2:*.COM"
will unprotect all files on drive 1 which have a filename extension of ".COM".

XIO 36,#1,0,0,"D1:DOS.SYS"
will unprotect only the file named "DOS.SYS" on the diskette in drive 1 (this step is necessary before erasing that file, as you might do to gain more space on the diskette).

8.16 How Do You Automatically Run BASIC Programs When DOS XL is Booted?

DOS XL is capable of booting directly into a BASIC program. In order to do

so, you must perform a few simple operations, which are presented in step-by-step fashion below:

Step 1: Boot a DOS XL master diskette. If the diskette you booted automatically executes the DOS XL menu, select the "Quit to DOS XL" menu item.

Step 2: If you want the BASIC program to start running when you boot another diskette rather than the one you just booted, place the other diskette in the drive at this time. NOTE: If the diskette is not initialized, refer to "How Do You Initialize a Disk From Command Level?".

Step 3: Type "TYPE E: STARTUP.EXC" followed by the <RETURN> key. DOS XL will blank out the screen and then the cursor will appear at the upper left corner of the screen. In the following line which you will type, the "basprog" part is a fictitious file name which should be replaced by the name of the BASIC program you wish to be executed. Type: DO CAR;RUN"D:basprog" followed by the <RETURN> key. NOTE: The double quotes (") should be typed as part of the line for this particular line. Also NOTE: whichever BASIC program you specify in substitution for "basprog" must be SAVEd onto this diskette before trying to boot this diskette. Now hold down the <CONTROL> or <CTRL> key and then type "3". This tells DOS XL that you are finished typing stuff to be placed into the file STARTUP.EXC. NOTE: This file can also be created from the DOS XL menu by Copying from E: to D:STARTUP.EXC and using the same CTRL-3 to indicate when you've completed the E: entry. NOTE: See "What Does the STARTUP.EXC File Do?" and "Controlling DOS XL Without Being There" for an explanation of the STARTUP.EXC file you have just created.

Step 4: Type "DIR" followed by the <RETURN> key. Check the "DIR" listing to make sure all of the following files are listed: DOS.SYS, DO.COM, STARTUP.EXC, and the BASIC program which you specified in the line you typed above. If any of these files are missing, your diskette will not boot correctly. If DO.COM is missing, see "How Do You Copy Files From Command Level" for an explanation of how to copy it from your DOS XL master to this diskette. If your BASIC program is on a different diskette, get into Atari cartridge BASIC and "LOAD" the program from the diskette it is on and then "SAVE" it to this diskette (see "How Do You Go to a Cartridge From Command Level" and "Using DOS XL with the Atari BASIC Cartridge"). DOS.SYS should have been placed onto the diskette when you

initialized the diskette, and STARTUP.EXC should be there since you just typed it in the previous step.

Step 5: Last, but not least, before trying to re-boot using this newly created diskette, make sure your Atari BASIC cartridge is installed correctly.

Step 6: Turn your Atari computer's power off and then on again to re-boot using this new diskette.

Section 9: CONTROLLING DOS XL WITHOUT BEING THERE

9.1 So What is DOS XL Batch Processing Anyway?

Once you've become experienced using DOS XL and are actually using it to perform some serious tasks for you, you may often find yourself repeating the same group of Command Processor commands over and over. In such a case, you can place these commands into a "batch" or "execute" file; and then later tell DOS XL, in a single quick command, to execute all the commands which you have placed into this file. This can save quite a bit of your time and energy since you won't constantly be typing the same sequence of commands.

Let's suppose that you wrote several programs that had to be run in sequence. Without using a batch file, you could issue the Command Processor commands necessary to run each program one at a time. If each program in the sequence took a long time to run, you would probably find yourself sitting at the keyboard for hours waiting for each program to finish just so you could type the command(s) necessary to get the next program running.

But, since you'ld probably rather go watch television, take a nap, or something while your Atari does all your work for you (what are computers for, anyway?), having a batch file issue those commands to DOS XL when necessary would probably be more to your liking.

All you would need to do in order for you to go off and have fun while your computer slaves away is to create a batch file containing the DOS XL commands required to run your set of programs. You would then enter the one command which tells DOS XL to take the rest of its commands from that batch file while you go off for your nap.

Any text file with the filename extension ".EXC" can be used as a DOS XL batch execute file. You tell DOS XL to start taking its commands from your batch file in much the same way as you give DOS XL an extrinsic command. The only difference is that you preceed the batch file's name (do not include the ".EXC" extension) with the commercial "at" symbol ("@"). For example, if you had already prepared a batch file called "DEMO.EXC" on drive one, and your default drive was on, you would type:

```
@DEMO<RETURN>
```

to tell DOS XL to begin taking its commands from the file "DEMO.EXC". When you give this command to DOS XL, the Command Processor will attempt to open the file "D1:DEMO.EXC" and then set up DOS XL to read it line by line, executing the Command Processor commands contained within the file just as if you were typing them from the keyboard.

9.2 What Is Contained in a DOS XL Batch ".EXC" File?

An "execute" or "batch" file is simply a text file. Each line of this text file will become a command to the Command Processor when you tell the Command Processor to "execute" the batch file.

There are just three simple rules you must follow when you are creating each line of the batch file:

1) The line must contain a valid DOS XL Console Processor command.

2) The line must be shorter than 128 characters in length.

3) The line must end in a standard Atari carriage return character (ATASCII $9B hexadecimal, see your Atari manual).

DOS XL allows the commands in an execute file to be preceeded by numbers and blanks. This feature allows the command lines to be numbered for readability and to document their purposes. This means that the command file lines:

```
LOAD OBJ.TEST<RETURN>
```

and:

```
100 LOAD OBJ.TEST<RETURN>
```

are the same to DOS XL. The Command Processor will search the line for the first non-numeric, non-blank character before starting to examine the command itself. Almost any text editor, including program editors supplied with many languages (such as MAC/65 from Optimized Systems Software), can be used to

create and modify execute files.

PROBLEM PREVENTOR: It is not possible to use the Atari BASIC cartridge's program editing mode to edit an execute file.

SPECIAL NOTE: You may also create an execute file (or, for that matter, any text file) by using the Command Processor intrinsic command "TYP E: <file-name>". The TYPe command will clear the screen, at which time you simply type in your text, line by line. When you are finished typing all the lines of your text/execute file, you hold down the <CONTROL> or <CTRL> key and type the "3" key in order to tell DOS XL you are finished. Although this approach will work, it is difficult to use since, once you press <RETURN> at the end of a line, you cannot go back to that line and make any changes. Although the cursor movement keys on your Atari keyboard will appear to function correctly while you enter your text, you can only make corrections to the current line you are typing without making a mess of your text file.

9.3 What Commands are Special for Batch Files?

DOS XL has four special intrinsic commands designed for use exclusively with execute files. These commands are:

REMARK	Remark or comment (does nothing)
SCREEN	Turn on the echoing of execute file command lines to the screen. (This is the default.)
NOSCREEN	Turn off echoing of execute file command lines.
END	Stop executing the execute file and return DOS XL to keyboard entry mode.

A more detailed discussion of these commands is provided under "The DOS XL Command Processor" sections of this manual.

9.4 How Is a Batch File Stopped?

While an execute file is being processed, various conditions may occur which will warrant a halt in the batch execution. These conditions may occur because of system-detected errors or because of a user program detecting a condition it considers hazardous to the system's health.

9.4.1 ...by DOS XL?

Humans are not quite perfect in the eyes of computers and sometimes make mistakes. If DOS XL detects an error in a human specified command, DOS XL will respond with an error message. If DOS XL discovers such an error while processing an execute file, it will print the error message as usual and then stop executing the execute file. Note that this stopping due to an error only occurs if the error is found by DOS XL, not if a program detects an error.

Execution of an execute file will also stop after the CARtridge commmand is executed.

Finally, execution of course stops when the end of the execute file is reached since there are no more commands to be taken from the file.

9.4.2 ...by a Program?

PROBLEM PREVENTOR: The following discussion is intended for assembly/machine lanuage, BASIC, and/or other language programmers. If you do not use your Atari for programming, or if the programs you write have no need to stop a batch file from executing, then you need not understand the following section.

It is sometimes desirable for a program in a chain of executing programs to stop the execute process. The usual reason for this is that the program has detected an error severe enough that the programs which follow it in the chain should no longer be executed. There is a flag kept in a special memory location by DOS XL which tells DOS XL that it is supposed to be taking its commands from an execute file. If a program sets this special memory location to the value zero, then DOS XL will no longer "remember" it is taking its commands from an execute file and will therefore resume taking commands from the keyboard. The execute flag is located 12 bytes from the starting location

of DOS XL within your Atari's memory. The starting location of DOS XL is contained within the 6502 word memory location 10 ($0A). The following BASIC program segment will turn off the execute file and return to DOS XL:

```
1000DOSLOC=PEEK(11)*256+PEEK(10) : REM DOSLOC=DOS'SSTART ADDRESS
1010 EXCFLG = DOSLOC+11 : REM EXCFLG = EXECUTE FLAG'S ADDRESS
1020 POKEEXCFLG,0 : REM SET DOS'S EXECUTE FLAG TO ZERO
1030 DOS : REM RETURN TO DOS XL
```

Or, from Optimized Systems' BASIC XL, you could simply use

```
100 POKE DPEEK(10)+11,0 : DOS
```

(Remember, though, that a CARtridge command automatically stops ".EXC" file execution, so this example may not be useful from BASIC.)

9.5 What is Special About the STARTUP.EXC Batch File?

The execute file "STARTUP.EXC" has special meanings to DOS XL. When DOS XL is first booted (when you turn on your Atari), DOS XL will search the directory of the booted diskette for a file named STARTUP.EXC. If STARTUP.EXC is on the booted diskette, DOS XL will execute that file before requesting keyboard commands.

A more detailed discussion of the "STARTUP.EXC" file is provided under "What Does The STARTUP.EXC File Do?".

9.6 How Do Execute Files Work?

PROBLEM PREVENTOR: The following section deals with programmer subject matter and may not be suitable for persons who still retain their sanity. As a result, Indus Systems recommends that if sane (non-programmer) people are still reading this section at this point, they should skip the remainder of this section. Thank you.

When you type in the Command Processor command "@<file-name>", the Command

Processor actually stores the <file-name> in an internal storage location (CPEXFN) and sets a flag (CPEXFL) to indicate that a batch operation is in progress.

Each time the Command Processor prompts the user (with "D1:", "D2:", etc.) it checks this flag to see if a batch execution is active. If so, it opens the batch file (using the stored <file-name>). Unless this is the first time the batch file has been opened (again, kept track of via a bit in CPEXFL), the Command Processor POINTs to the start of the next text line in the file.

The next text line is then read into the command buffer. Then the Command Processor NOTEs the new position in the file and saves the position (CPEXNP) for use by the next needed POINT process (as above).

Finally, the command in the command buffer is executed just as if the user had typed it from the keyboard. If the command properly terminates (via an RTS or a JMP through DOSVEC), the entire process repeats, until the execute flag is somehow turned off.

The experienced programmer will no doubt realize that changing the contents of the various CPEXxx locations can affect batch execution in possibly very interesting ways. These locations are all defined in the file called SYSEQU.ASM and are offsets from the address contained in DOSVEC (location $000A). See, as an example, the sample program segment given above which stops the execution of a batch file.

Some of this information is further discussed under "Using DOS XL with Assembly Language".

Section 10: USING DOS XL WITH ASSEMBLY LANGUAGE

PROBLEM PREVENTOR: This entire section contains extremely technical information relating to the interfacing of assembly/machine language programs to DOS XL. If you are not an assembly/machine language programmer, you will have absolutely no need in the world to understand the following discussion. If you are a beginning assembly/machine language programmer, this is not the place to start when learning assembly/machine language. All of the following information is intended for the experienced assembly language programmer, and no attempt has been made to "soften" the discussion in order to accomodate users who have no experience with the type of information being discussed.

SPECIAL NOTE: The information contained in this section has been supplied by Indus Systems as a courtesy to those users who "wanted to know". Unfortunately, this information relates to performing operations which are far beyond the needs of the vast majority of DOS XL users; and therefore any questions arising from the use of the information contained in this section would be in the extreme minority. Because of the unbalanced nature of normal user support versus technical/advanced user support, very little support can be provided by Indus for those users who have trouble with the information contained in this section. Should you encounter problems with the information in this section, Indus will attempt to assist you as best as we can; but in general you should consider yourself "on your own".

DOS XL is designed as a layered operating system. Application programs (including languages such as BASIC XL) are expected to call the operating system "properly", through the system call vector (labeled "CIO" in the file "SYSEQU.ASM" on your DOS XL System Master diskette). In turn, the CIO will determine which device is to receive what I/O request and handles most of the work transparent to the calling program.

If a program restricts itself to proper calls to CIO using labels provided in SYSEQU.ASM, the program should transfer virtually without change from one version of DOS XL to another. (Probably the only other areas of change would involve memory map usage.)

In any case, herewith is a description of the proper assembly language calling sequences and parameters under DOS XL.

10.1 Interfacing to I/O Routines

10.1.1 The Structure of the IOCB's

When a program calls the OS through location "CIO", OS expects to be given the address of a properly formatted IOCB (Input Output Control Block). For simplicity, we have predefined 8 IOCB's, each 16 bytes long, and the calling program specifies which one to use by passing the IOCB number times 16 in the 6502's X-register. Thus, to access IOCB number four, the X-register should contain $40 on entry to OS. Notice that the IOCB number corresponds directly to the file number in BASIC (as in PRINT #6, etc.). The IOCB's are located from $0340 to $03BF on the Atari (but you really should use the equates from the disk file "SYSEQU.ASM" rather than relying on hard-coded addresses.)

When the OS gets control, it uses the X-register to inspect the appropriate IOCB and determine just what it was that the user wanted done. Figure 10-1 gives the DOS XL standard name for each field in the IOCB along with a short description of the purpose of the field. Study the figure before proceeding.

The user program should never touch fields ICHID, ICDNO, ICSTA and ICPUT, as they are set by the OS. In addition, unless the particular device and I/O request requires it, the program should not change ICAUX1 through ICAUX6. The most important field is the one-byte command code, ICCOM, which tells the operating system what function is desired.

IOCB Structure

Field Name	Offset Within IOCB (bytes)	Size of Field (bytes)	Purpose of Field
ICHID	0	1	Set by OS. Index into device name table for currently open file, set to $FF if no file open on this IOCB.
ICDNO	1	1	Set by OS. Device number (e.g., 1 for "D1:xxx" or 2 for "D2:yyy")
ICCOM	2	1	The command request from user program. Defines how rest of IOCB is formatted.
ICSTA	3	1	Set by OS. Last status returned by device. Not necessarily the status returned via STATUS command request.
ICBADR	4	2	Buffer Address. A two byte address in normal 6502 low/high order. Specifies address of buffer for data transfer or address of filename for OPEN, STATUS, etc.
ICPUT	6	2	Set by OS. Address minus one of device's put-one-byte routine. Possibly useful when high speed single byte transfers are needed.
ICBLEN	8	2	Buffer length. Specifies maximum number of bytes to transfer for PUT/GET operations. Note: this length is decremented by one for each byte transfered.
ICAUX1	10	1	Auxiliary byte number one. Used in OPEN to specify kind of file access needed. Some drivers can make additional use of this byte.

Field Name	Offset Within IOCB (bytes)	Size of Field (bytes)	Purpose of Field
ICAUX2	11	1	Auxilliary byte number two. Some serial port functions may use this byte. This and all following AUX bytes are for special use by each device driver.
ICAUX3	12	2	For disk files only: where the ICAUX4 disk sector number is passed by NOTE and POINT. (These bytes could be used separately by other drivers.
ICAUX5	14	1	For disk files only: the byte-within-sector number passed by NOTE and POINT.
ICAUX6	15	1	A spare auxilliary byte.

IOCB Field Usage
Standard Commands

IOCB Field --> / Name	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	IOCB <--- Field Name (as used in SYSEQU.ASM)
	I/O	I/O	I/O	I/O	Buf Adrs	Put-a-Byte Adrs		Buf Len			I/O	I/O	I/O	I/O	I/O	I/O	
Type of command	HI/DO	NI/O	OI/M	TI/A	ICB-ADR	ICPUT	ICB-LEN				UI/XI	UI/XI	UI/XI	XI/XI	XI/XI	XI/XI	COMMAND NAMES
OPeN	*	*	3	*	Fname	*					md						COPN
CLOSE	*		12	*													CCLOSE
dynamic STATus		*	13	*	Fname												CSTAT
Get TeXT Record			5	*	Buf			len									CGTXTR
Put TeXT Record			9	*	Buf			len									CPTXTR
Get BINary Record			7	*	Buf			len									CGBINR
Put BINary Record			11	*	Buf			len									CPBINR

(table continued)

IOCB Field Usage (continued)
Extended Commands: Disk File Manager Only

IOCB Field --> Name	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	IOCB <--- Field Name (as used in SYSEQU.ASM)
	I/O command	I/O command	I/O command	I/O command	Buf Adrs		Put-Byte Adrs		Buf Len		C/A/U/ 1	C/A/U/ 2	C/A/U/ 3	C/A/U/ 4	C/A/U/ 5	C/A/U/ 6	
	Type of command	H/I/D/	D/I/N/O	C/O/M	S/T/A	ICB-ADR		ICPUT	ICB-LEN		X	X	X	X	X	X	Command Names
RENname			*	32	*	Fname											CREN
ERAse			*	33	*	Fname											CERA
PROtect			*	35	*	Fname											CPRO
UNProtect			*	36	*	Fname											CUNP
NOTE				38	*								secno		by		CNOTE
POINT				37	*								secno		by		CPOINT

LEGEND:
- '*' Set by OS when this command is used.
- 'Buf' Address of a data buffer.
- 'Fname' Address of a filename.
- 'len' Length of a data buffer.
- 'md' Mode of OPEN (read, write, etc.).
- 'secno' Sector number (see "The I/O Commands").
- 'by' Byte in sector (see "The I/O Commands").

10.1.2 The I/O Commands

The preceeding tables provide a summary of the I/O commands and the associated usage of the various fields of the IOCB's. The first seven are DOS XL oriented and will be dealt with in "The Standard DOS XL Commands" discussion below. The last six are File Manager specific and are discussed in "Commands Unique to the Disk File Manager System".

Most of the commands manipulate a device in some way, so maybe we should talk about them for a moment. Device names under DOS XL are very simplistic; they consist of a single letter optionally followed by a single digit used to define a specific device when more than one of the same kind exist (for example, D1: or D2:). Traditionally (and, in the case of Atari disk files, of necessity) the device name is followed by a colon. The following devices are implemented under standard DOS XL and Atari DOS:

E: The keyboard/screen editor device. The normal console output.

K: The keyboard alone. Use this device to bypass editing of user input.

S: The screen alone. Can be either characters (ala E:) or graphics.

P: On the Atari, the printer. The standard device driver allows only one printer.

C: The cassette recorder.

D: The disk file manager, which also usually requires a file name.

Other device names are possible (for RS-232 interfaces, for example), and in fact the ease with which other devices may be added is another mark for the claim that DOS XL is a true operating system. (The structure of device drivers is material for a later section.) But we should like to point out that, on the Atari, the OS ROM includes drivers for all the above except the disk. In fact, the drivers account for over 5K bytes of the ROM code. The screen handler, with all its associated editing and GRAPHICS modes, occupies about 3K bytes of that.

10.1.2.1 The Standard DOS XL Commands

The OS itself only understands a few fundamental commands, but DOS XL also provides for the extended commands necessary to some devices (XIO in BASIC). In any case, each of these fundamental commands deserves a short description.

10.1.2.1.1 OPEN

Open a device (synonyms: file, IOCB, channel) for read and/or write access. OS expects ICAUX1 to contain a byte that specifies the mode of access:

ICAUX1	MODE
4	Read Only
6	Read Directory Only
8	Write Only
9	Write OnlyAppend
12	Read/Write (Update)

The name of the device (and, for the disk, the file) must be given to OS; this is accomplished by placing the ADDRESS of a string containing the name in ICBADR.

10.1.2.1.2 CLOSE

Terminate access to a device/file. Only the command must be given.

10.1.2.1.3 STATUS

Request the status of a device/file. The device can interpret this request as it wishes, and pass back a (hopefully) meaningful status. As with OPEN, the ADDRESS of a filename must be placed in ICBADR.

10.1.2.1.4 GET TEXT

A powerful command, this causes the OS to retrieve ("GET") bytes one at a time from a device/file already OPENed until either the buffer space provided by the user is exhausted or a RETURN character (Atari $9B) is encountered. The user specifies the buffer to use by placing its ADDRESS in ICBADR and its maximum size (length) in ICBLEN.

10.1.2.1.5 PUT TEXT

The analogue of GET TEXT, OS outputs characters one at a time until a RETURN is encountered or the buffer is empty. Requires ICBADR and ICBLEN to be specified.

10.1.2.1.6 GET DATA

Extremely flexible command, this causes OS to retrieve, from the device/file previously OPENed, the number of bytes specified by ICBLEN into the buffer specified by ICBADR. No checks what-so-ever are performed on the contents of the transferred data.

10.1.2.1.7 PUT DATA

Similar to GET DATA, except that OS will output ICBLEN bytes from the buffer specified by ICBADR . Again, no data checks are performed.

10.1.2.2 Commands Unique to the Disk File Manager System

The second part of the "IOCB Field Usage" tables shows several DOS XL system commands not yet discussed. These "extended" commands are accessed via the extended request routine in a device driver's handler table (see "Device Handlers" for details on device drivers). However, some of these extended

commands as implemented for the disk device in the File Manager System are important enough to deserve their own sections. We'll examine each of the extended disk operations in a little detail:

10.1.2.2.1 ERASE, PROTECT, and UNPROTECT

Also known as Delete, Lock, and Unlock, these three commands simply provide OS with a channel number (i.e., the X-register contains IOCB number times 16), a command number (ICCOM), and a filename (via ICBADR). When OS passes control to the FMS, an attempt is made to satisfy the request. Note that the filename may include "wild cards", as in "D:*.??S" (which will affect all files on disk drive one which have an 'S' as the last letter of their filename extension).

10.1.2.2.2 RENAME

Very similar to ERASE, et al, in usage. The only difference is in the form of the filename. Proper form is: "[Dn:]oldname.ext,newname.ext" Note that the disk device specifier is not and CAN NOT be given twice.

10.1.2.2.3 NOTE and POINT

Other than OPEN, these are the only commands encountered in standard DOS XL which use any of the AUXilliary bytes of the IOCB. For these commands, the user specifies the channel number and command number and then receives or passes file pointer information via three of the AUX bytes. ICAUX3/ICAUX4 are used as a conventional 6502 LSB/MSB 16-bit integer: they specify the current (NOTE) or the to-be-made-current (POINT) sector within an already OPENed disk file. ICAUX5 is similarly the current (NOTE) or to-be-made-current (POINT) byte within that sector.

10.1.2.3 FMS Extensions of the OPEN Command

Open is not truly an extended operation, but for disk I/O we need to know that the FMS allows two additional "modes" beyond the fundamental OS modes.

If ICAUX1 contains a 6 when DOS XL is called for OPEN, then the disk directory is opened (instead of a file) for read-only access. The address ICBADR now specifies the file (or files, if wild cards are used) to be listed as part of a directory listing. Note that FMS expects this type of OPEN to be followed by a succession of GETREC (get text line) OS calls.

If ICAUX1 contains a 9, the specified file is opened as a write-only file, but the file pointer is set to the current end-of-file.

10.1.3 Error Codes Returned

On return from any OS call, the Y-register contains the completion code of the requested operation. A code of one (1) indicates "normal status, everything is okay". (I know, why not zero, which is easier to check for. Remember, we based this on Atari's OS ROMs, which are good, not perfect.) By convention, codes from $02 to $7F (2 through 127 decimal) are presumed to be "warnings". Those from $80 to $FF (128 through 255 decimal) are "hard" errors. These choices facilitate the following assembly language sequence:

```
JSR  CIOV      ; call the OS
TYA            ; check error code
BMI  OOPS      ; if $80-$FF, it must be an error
```

In theory, DOS XL always returns to the user with condition codes set such that the TYA is unnecessary. In practice, that's probably true; but a little paranoia often leads to longer life of both humans and programs.

10.2 Manipulation of DOS XL

The writer of assembly language code will most likely need to interface with the Atari Operating System (OS) in some way. If the assembly code is to become an extrinsic command, there may be a need to interface to DOS XL. See "Interface to I/O Routines" for more information about the OS interface.

If you are writing software designed to interface with DOS XL, you may need to examine and/or modify certain special memory locations or access certain routines within DOS XL. This section lists and describes those that we feel are the most useful.

10.2.1 SYSEQU.ASM

Every DOS XL master disk contains an assembler source file, SYSEQU.ASM, that has various commonly used Atari OS and DOS XL system equates. This file may be included in an assembly language program via the OSS MAC/65 include function (.INCLUDE #D1:SYSEQU.ASM); however, it exists on the master disk as a text file and must be 'ENTER'ed into MAC/65 and then 'SAVE'ed back to the disk.

10.2.2 CP MEMORY LOCATIONS

The Command Processor (CP) on the Atari is designed to be placed just after the normal Atari File Manager when the DOS.SYS version of DOS XL is used. Since the actual location of CP may vary with different versions of the file manager and/or because of different memory configurations, a fixed location has been assigned to point to CP. The location CPALOC ($0A on the Atari) contains the address of the DOS XL and CP warmstart entry point. Most Atari programs should return to CP by JMPing to the address contained in CPALOC.

10.2.3 EXECUTE PARAMETERS

The CP execute flag is located CPEXFL ($0B) from the start of CP. The CPALOC may be used as an indirect pointer to access the execute flag:

```
LDY  #CPEXFL        ;GET DISPL TO FLAG
LDA  (CPALOC),Y     ;LOAD FLAG
```

The Execute Flag has four bits that control the execute process:

Name	Bit #	Description
EXCYES	$80	If one, an execute is in progress
EXCSCR	$40	If one, do not echo execute input to screen
EXCSUP	$20	If one, a cold start execute is starting. Used to avoid a FILE NOT FOUND error if STARTUP.EXC is not on boot disk.
EXCNEW	$10	If one, a new execute is start-ing. Tells CP to start with the first line of the file

CP performs the execute function by OPENing the file, POINTing to the next line, READing that line, NOTEing the new next line and CLOSEing the file. To perform these functions, CP must save the execute file name and the three byte NOTE values. The filename is saved at CPEXFN ($0C) into CP. The three NOTE values are saved at CPEXNP ($1C) into CP. (CPEXNP = ICAUX5; CPEXNP + 1 = ICAUX4; CPEXNP + 2 = ICAUX3). By changing the various execute control parameters, a programmer can cause chaining of execute files, skipping of certain lines in the file, etc.

10.2.4 DEFAULT DRIVE LOCATION

The CP default drive file spec is located at CPDFDV ($07) into OS/A+. The Default Drive here is ATASCII Dn: where "n" is the ATASCII default drive number.

10.2.5 EXTRINSIC PARAMETERS

The extrinsic commands may be called with parameters typed on the command line. The CP command:

```
D1:COPY FROMFILE D2:TOFILE
```

is an example of this. The entire command line is saved in the CP input buffer located at CPCMDB ($3F) bytes into CP and is available to the user. Since most command parameters are file names, CP provides a means of extracting these parameters as filenames. The routine that performs this service begins at CPGNFN ($03) bytes into CP . The routine will get the next parameter and move it to the filename buffer at CPFNAM ($21) bytes in CP. If the parameter does not contain a device prefix, then CP will prefix the parameter with the default drive prefix. The first time COPY calls CPGNFN the file spec "D1:FROMFILE" is placed at CPFNAM. The second time COPY calls CPGNFN the file spec "D2:TO FILE" is placed in CPFNAM. If CPGNFN were to be called more times, then the default file spec would be set into CPFNAM at each call. To detect the end of parameter condition, the user may check the CPBUFP ($0A into CP) cell. If CPBUFP does not change often a CPGNFN call then there are no more parameters. The filename buffer is always padded to 16 bytes with ATASCII EOL ($9B) characters. The following example sets up a vector for calling the get file name routine:

```
            CLC
            LDA  CPALOC     ;ADD CPGNFN
            ADC  #CPGNFN    ;TO CPALOC VALUE
            STA  GETFN+1    ;AND PLACE IN
            LDA  CPALOC+1   ;ADDRESS FIELD
            ADC  #0         ;OF JUMP
            STA  GETFN+2    ;INSTRUCTION
GETFN       JMP  0
```

The following routine gets the next file name to CPFNAM:

```
    LDY  #CPBUFP        ;SAVE CPBUFP
    LDA  (CPALOC),Y     ;VALUE
    PHA
    JSR  GETFN          ;GET NEXT FILE PARM
    LDY  #CPBUFP
    PLA                 ;TEST FOR NO NEXT
    CMP  (CPALOC),Y     ;PARM
    BEQ  NONEXT         ;BR IF NO NEXTPARM
    LDY  #CPFNAM        ;ELSE GET FILE
    LDA  (CPALOC),Y     ;NAME FROM BUFFER
```

10.2.6 RUNLOC

Whenever an extrinsic command is invoked, RUNLOC ($3D into CP) is given the value of the first address in that command's .COM file. Some extrinsic commands (including user written commands) can therefore be restarted by typing the RUN command. You may want to change the contents of RUNLOC to point to the warmstart point of your program when it's entered the first time to avoid unwanted reinitializations when re-entered. BASIC A+ and MAC/65 do this to avoid clearing any user program which may be in memory when returning from CP. If you want to forbid re-entry, you need to set RUNLOC's high order byte ($3E into CP) to zero:

```
LDY  #RUNLOC+1       ;FORBID RE-ENTRY
LDA  #0              ;TO ME
STA  (CPALOC),Y
```

10.3 DEVICE HANDLERS

As we have noted before, CIO is actually a very small program (approximately 700 bytes). Even so, it is able to handle the wide variety of I/O requests detailed in the earlier parts of this section with a surprisingly simple and consistent assembly language interface. Perhaps even more amazing is the purity and simplicity of the OS interface to its device handlers.

Admittedly, because of this very simplicity, CIO is sometimes slower that one would wish (only noticeably so with PUT BINARY RECORD and GET BINARY RECORD) and the handlers must be relatively sophisticated. But not too much so, as we will show.

10.3.1 The Device Handler Table

At location "HATABS" in RAM, CIO has (loaded from ROM on the Atari) a list of the standard devices (P:, D:,E:,S:, and K:) and the addresses thereof. To add a device, simply tack it on to the end of the list: you need only specify the device's name (one character) and the address of its handler table (more on

that in a moment).

In theory, all named device handlers under DOS XL may handle more than one physical device. Just as the disk handler understands "D1:" and "D2:", so could a keyboard handler understand "K1:" and "K2:". DOS XL supplies a default sub-device number of "1" if no number is given (thus "D:" becomes "D1:").

Following is the layout of the HAndler TABleS on the Atari computers:

```
            *=          $031A
HATABS
            .BYTE       'P'         ; the Printer device
            .WORD       PDEVICE     ; and the address of its driver
            .BYTE       'C'         ; the Cassette device
            .WORD       CDEVICE
            .BYTE       'E'         ; the screen Editor device
            .WORD       EDEVICE
            .BYTE       'S'         ; the graphics Screen device
            .WORD       SDEVICE
            .BYTE       'K'         ; the Keyboard device
            .WORD       KDEVICE
            .BYTE       0           ; zero marks the end of the table
            .WORD       0           ; ...but there's room for serveral
            .BYTE       0           ; ...more devices
        ...et cetera...
```

10.3.2 Rules for Writing Device Handlers

Each device which has its handler address placed into the handler address table (above) is expected to conform to certain rules. In particular, the driver is expected to provide six (6) action subroutines and an initialization routine. (In practice, the current Atari's OS only calls the initialization routines for its own pre-defined devices. Since this may change in the future, and since one can force the call to one's own initialization routine, we must recommend that each driver include one, even if it does nothing.) The address placed in the handler address table must point to, again, another table, the form of which is shown below (Figure 10.3).

```
HANDLER
     .WORD     <address of OPEN routine>-1
     .WORD     <address of CLOSE routine>-1
     .WORD     <address of GETBYTE routine>-1
     .WORD     <address of PUTBYTE routine>-1
     .WORD     <address of STATUS routine>-1
     .WORD     <address of XIO routine>-1
     JMP       <address of initialization routine>
```

Notice the six addresses which must be specified; and note that in the table one must subtract one from each address (the "-1" simply makes CIO's job easier...honest). A brief word about each routine is given in the following pages.

10.3.2.1 Device OPEN

The OPEN routine must perform any initialization needed by the device. For many devices, such as a printer, this may consist of simply checking the device status to insure that it is actually present. Since the X-register, on entry to each of these routines, contains the IOCB number being used for this call, the driver may examine ICAUX1 (via LDA ICAUX1,X) and/or ICAUX2 to determine the kind of OPEN being requested. (Caution: CIO preempts bits 2 and 3 ($04 and $08) of ICAUX1 for read/write access control. These bits may be examined but should normally not be changed.)

10.3.2.2 Device CLOSE

The CLOSE routine is often even simpler. It should "turn off" the device if necessary and possible.

10.3.2.3 Device PUT and GET BYTE Routines

The PUTBYTE and GETBYTE routines are just what are implied by their names: the

device handler must supply a routine to output one byte to the device and a routine to input one byte from the device. HOWEVER, for many devices one or the other of these routines doesn't make sense (ever tried to input from a printer?). In this case the routine may simply RTS and DOS XL will supply an error code.

10.3.2.4 Device STATUS Routine

The STATUS routine is intended to implement a dynamic status check. Generally, if dynamic checking is not desirable or feasible, the routine may simply return the status value it finds in the user's IOCB. However, it is NOT an error under DOS XL to call the status routine for an unOPENed device, so be careful.

10.3.2.5 Device Extended I/O Routine(s)

The XIO routine does just what its name implies: it allows the user to call any and all special and wonderful routines that a given device handler may choose to implement. OS does nothing to process an XIO call except pass it to the appropriate driver.

10.3.2.6 General Comments on Device I/O Routines

In general, the AUXilliary bytes of each IOCB are available to each driver. In practice, it is best to avoid ICAUX1 and ICAUX2, as several BASIC and OS commands will alter them to their will. Note that ICAUX3 thru ICAUX5 may be used to pass and receive information to and from BASIC via the NOTE and POINT commands (which are actually special XIO commands). Finally, drivers should not touch any other bytes in the IOCBs, especially the first two bytes.

Notice that handlers need not be concerned with PUT BINARY RECORD, GET TEXT RECORD, etc.: OS performs all the needed housekeeping for these user-level commands.

10.3.3 Rules for Adding Things to OS

1. Inspect the system MEMLO pointer (see SYSEQU.ASM for the actual location).

2. Load your routine (including needed buffers) at the current value of MEMLO.

3. Add the size of your routine to MEMLO.

4. Store the resultant value back in MEMLO.

5. Connect your driver to OS by adding its name and address into the handler address table.

6. Fool OS so that if SYSTEM RESET is hit steps 3 thru 5 will be reexecuted (because SYSTEM RESET indeed resets the handler address table and the value of MEMLO). In point of fact, step 2 is the hardest of these to accomplish. In order to load your routine at wherever MEMLO may be pointing, you need a relocatable (or self-relocatable) routine. Since there is currently no assembler for the Atari computers which produces intrinsically relocatable code, this is not an easy task. But it may not be necessary if you are writing code for your own private system instead of the general public.

Step 6 is accomplished by making Atari OS think that your driver is the Disk driver for initialization purposes (by "stealing" the DOSINI vector) and then calling the Disk's initializer yourself before steps 3 thru 5 are performed again.

10.3.4 An Example Program

This driver, included in source form on some DOS XL distribution diskettes as "MEM.LIS", builds a new driver and adds it to the operating system. The "device" being driven is simply excess system memory within your computer. Thus, you may (for example) use this as a pseudo-disk file for passing data between sequentially called programs.

Some words of caution are in order. This driver does not perform step 6 as noted in the last section (but it may be reinitialized via a BASIC USR call). It does not perform self-relocation: instead it simply locates itself above all normal low memory usage (except the serial port drivers, which would have to be loaded after this driver). If you assemble it yourself, you could do so at the MEMLO you find in your normal system configuration (or you could improve it to be self-modifying, of course).

Other caveats pertain to the handler's usage: it uses RAM from the contents of MEMTOP downward. It does not check to see if it has bumped into BASIC's MEMTOP ($90) and hence could conceivably wipe out programs and/or data. To be safe, don't write more data to the RAM than a FRE(0) shows (and preferrably even less).

In operation, the M: driver reinitializes upon an OPEN for write access (mode 8). A CLOSE followed by a subsequent READ access will allow the data to be read in the order it was written.

More Cautions: don't change graphics modes between writing and reading if the change would use more memory (to be safe, simply don't change at all). The M: will perform almost exactly as if it were a cassette file, so the user program should be data sensitive if necessary: the M: driver will NOT itself give an error based on data contents. Note that the data may be re-READ if desired (via CLOSE and re-OPEN).

The following is a suggested set of BASIC programs:

Ending of PROGRAM 1:

```
9900 OPEN #2,8,0,"M:"
9910 PRINT #2; LEN(A$)
9920 PRINT #2; A$
9930 CLOSE #2
9940 RUN "D:PROGRAM2"
```

Beginning of PROGRAM 2:

```
100 OPEN #4,4,0,"M:"
110 INPUT #4,SIZE
120 DIM STRING$(SIZE)
```

```
130 INPUT #4, STRING$
140 CLOSE #4
```

BASIC XL users might find RPUT/RGET and BPUT/BGET to be useful tools here instead of PRINT and INPUT. And, of course, users of any other language(s) might find this a handy inter-program communications device.

Section 11: DISK FILE STRUCTURE

PROBLEM PREVENTOR: This entire section contains extremely technical information relating to DOS XL's maintainance of files on your diskettes. If you are not an advanced level programmer, you will have absolutely no need in the world to understand the following discussion. If you are a beginning level programmer, this is not the place to start when looking for some kind of practice program to write. All of the following information is intended for the experienced programmer, and no attempt has been made to "soften" the discussion in order to accomodate users who have no experience with the type of information being discussed.

SPECIAL NOTE: The information contained in this section has been supplied by Indus Systems as a courtesy to those users who "wanted to know". Unfortunately, this information relates to performing operations which are far beyond the needs of the vast majority of DOS XL users; and therefore any questions arising from the use of the information contained in this section would be in the extreme minority. Because of the unbalanced nature of normal user support versus technical/advanced user support, very little support can be provided by Indus for those users who have trouble with the information contained in this section. Should you encounter problems with the information in this section, Indus will attempt to assist you as best as we can; but in general you should consider yourself "on your own".

DOS XL version 2 was produced to provide the maximum compatibility possible with Atari's DOS 2.0s. In fact, the FMS used is identical to that used by Atari (for a simple reason: Optimized Systems wrote Atari's DOS). For reasons known best to Atari, we were instructed to create Atari's FMS around a linked-sector disk space management scheme. In essence, this means that the last three bytes of each sector in a disk file contain a link to the next sector in that same file. The positive result of this is that one produces a relatively small, memory-resident, disk manager which is nevertheless capable of dynamically allocating diskette space (unlike, for example, a contiguous file disk manager). The biggest disadvantage of the scheme seems to be that one may not do direct (random) access to the bytes of such files, as one can do with either a contiguous or mapped file allocation technique. Also, a disk error in the middle of a linked file means a loss of access to the rest of the file.

Section 11: DISK FILE STRUCTURE

The purpose of the FMS is to organize the 720 data sectors available on an Atari 810 (or its double density equivalent) diskette into a system of named data files. FMS has three primary data structures that it uses to organize the disk:

1. Volume Table of Contents (VTOC): a single disk sector which keeps track of which disk sectors are available for use in data files.

2. Directory: a group of eight contiguous sectors used to associate file names with the location of the files' sectors on the disk. Each Directory entry contains a file name, a pointer to the first data sector in the file, and some miscellaneous information.

3. Data Sectors: sectors containing the actual data and some control information that links one data sector to the next data sector in the file. Note: since double density diskette sectors contain 256 bytes whereas single density (810 drive) sectors contain only 128, certain absolute byte number references may vary depending upon the diskette in use. Throughout this chapter, in such cases, the single density number is given followed by the double density number in square brackets [thus].

11.1 Data Sectors

A Data Sector is used to contain the file's data bytes. Each 128 [256] byte data sector is organized to hold 125 [253] bytes of data and three bytes of control. The data bytes start with the first byte (byte 0) in the sector and run contiguously up to, and including, byte 124 [252]. The control information starts at byte 125 [253].

The sector byte count is contained in byte 127 [255]. This value is the actual number of data bytes in this particular sector. The value may range from zero (no data) to 125 [253] (a full sector). Any data sector in a file may be a short sector (contain less than 125 [253] data bytes).

The left six bits of byte 125 [253] contain the file number of the file. This number corresponds to the location of the file's entry in the Directory. Directory entry zero in Directory sector $169 has a file number of zero.

Entry one in Directory sector $169 has a file number one, and so forth. The file number value may range from zero to 63 ($3F). The file number is used to insure that the sectors of one file do not get mixed up with the sectors of another file.

The right two bits of byte 125 [253] (and all eight bits of byte 126 [254]) are used to point to the next data sector in the file. The ten bit number contains the actual disk sector number of the next sector. Its value ranges from zero to 719 ($2CF). If the value is zero then there are no more sectors in the file sector chain. The last sector in the file sector chain is the End-Of-File sector. The End-Of-File sector will almost always be a short sector.

11.2 Disk Directory

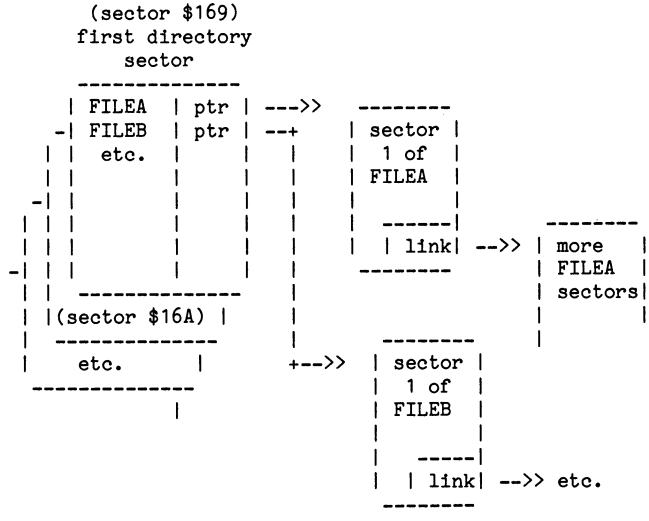
The Directory starts at disk sector $169 and continues for eight contiguous sectors, ending with sector $170. These sectors were chosen for the directory because they are in the center of the disk and therefore have the minimum average seek time from any place else on the disk. Each directory sector has space for eight file entries. Thus, it is possible to have up to 64 files on one disk.

A Directory entry is 16 bytes in size, as illustrated by the figure below. The directory entry flag field gives specific status information about the current entry. The directory count field is used to store the number of sectors currently used by the file. The last eleven bytes of the entry are the actual file name. The primary name is left justified in the primary name field. The name extension is left justified in the extension field. Unused filename characters are blanks ($20). The Start Sector Number field points to the first sector of the data file.

Directory Entry Structure

Starting Byte # of Field	Length of Field (bytes)	Purpose of Field
0	1	Flag byte. Meanings of bits: $00 Entry never used $80 Entry was deleted $40 Entry in use $20 Entry protected $02 a version 2 file $01 Now writing file
1	2	Count (LSB,MSB) of sectors in file
3	2	Start sector (LSB,MSB) of link chain
5	8	File name, primary
13	3	File name, extension

Directory Structure



Note: only eight file directory entries are stored per sector, even on double density diskettes.

11.3 Volume Table of Contents (VTOC)

The VTOC sector ($168) is used to keep track of which disk sectors are available for data file usage. The figure below illustrates the organization of the VTOC sector. The most important part of the VTOC is the sector bit map.

The sector bit map is a contiguous string of 90 bytes, each of which contains eight bits. There are a total of 720 (90 x 8) bits in the bit map--one for each possible sector on an 810 type diskette format. The 90 bytes of bit map

start at VTOC byte ten ($0A). The leftmost bit ($80 bit) of byte $0A respresents sector zero. The bit just to the right of the leftmost bit ($40 bit) represents sector one. The rightmost bit (bit $01) of byte $63 represents sector 719.

Structure of VTOC Sector

Starting Byte # of Field	Length of Field (bytes)	Purpose of Field
0	1	Reserved (for type code)
1	2	Total number of sectors
3	2	Number of unused sectors
5	5	Reserved
10	90	Sector usage bit map Each bit represents a particular sector: a 1 bit indicates an available sector, a 0 bit indicates a sector in use.
100	28	Reserved (could be used for version 2 type DOS with more than 720 sectors per disk)

Section 12: SYSTEM MEMORY MAP

PROBLEM PREVENTOR: This entire section contains extremely technical information relating to DOS XL's internal structure. If you are not an advanced level programmer, you will have absolutely no need in the world to understand the following discussion. If you are a beginning level programmer, this is not the place to start when looking for some kind of practice program to write. All of the following information is intended for the experienced programmer, and no attempt has been made to "soften" the discussion in order to accomodate users who have no experience with the type of information being discussed.

SPECIAL NOTE: The information contained in this section has been supplied by Indus Systems as a courtesy to those users who "wanted to know". Unfortunately, this information relates to performing operations which are far beyond the needs of the vast majority of DOS XL users; and therefore any questions arising from the use of the information contained in this section would be in the extreme minority. Because of the unbalanced nature of normal user support versus technical/advanced user support, very little support can be provided by Indus for those users who have trouble with the information contained in this section. Should you encounter problems with the information in this section, Indus will attempt to assist you as best as we can; but in general you should consider yourself "on your own".

12.1 Atari Zero Page Map

Location		Usage
0-9		System Zero Page
A-B	CPALOC	Known to Atari DOS as DOSVEC
C-D	DOSINI	Vector to FMS Initialization
E-42		System Zero Page
43-49		FMS Zero Page
4A-7F		System Zero Page
80-FF		User and Language Zero Page
80-CD		BASIC XL and Atari BASIC Zero Page
D2-FF		Floating Point Zero Page

12.2 Atari System Memory Map - DOS XL Version 2

Location		Usage
100-1FF		6502 Stack Area
200-319		System RAM
300-30B		DCB (Device Control Block)
31A-33F		Device Handler Table
340-3BF		IOCB's - 8 at 16 Bytes Each
3C0-57F		System RAM
580-5FF		E: Text Buffer
600-6FF		User RAM
700-varies		DOS XL -- File Manager and CP or Just Buffers, etc., When Using Extended Memory DOS Systems
709	SABYTE	Number of 128 Byte File Buffers
70A	DRVBYT	Bit Map: Accessible Drives
70C	SASA	Address of Start of Buffers
(2E7)-BFFF		User, Language, and Graphics Memory -- Note: (2E7) Means "Contents of Location $02E7" (LOMEM).
A000-BFFF		SuperCartridge and Atari BASIC Memory -- Also Used by DOS XL for File Manager and CP in "DOSXL.SUP" Version of the Extended Memory DOS System
C000-CFFF		Unused in Atari 400/800, OS ROM in XL-Series, Bank Switched With RAM
D000-D7FF		I/O Locations
D500-D5FF		Used by SuperCartridge for Bank Select
D800-DFFF		Floating Point ROM
E000-E3FF		Character Set ROM
E400-FFFF		OS Drivers, CIO, etc. -- in ROM
E400-FFF9		Bank-Selectable RAM Used by DOS XL for File Manager, CP, etc., in the "DOSXL.XL" Version of Extended Memory DOS

Section 13: ERRORS

13.1 Types of Errors

All DOS XL operations return a status value in the IOSTAT field. DOS XL convention is that status values of $80 or greater indicate some sort of error. There are four fundamental kinds of errors that can occur with DOS XL:

13.1.1 Hardware Errors

Such as attempting to read a bad disk, write a read-only disk, etc.

13.1.2 Data Transfer Errors

Errors which occur when data is transferred between the computer and a peripheral device. Examples include Device Timeout, Device NAK, Framing Error, etc.

13.1.3 Device Driver Errors

Found by the driver for the given device, as in (for the DFM) File Not Found, File Locked, Invalid Drive Number, etc.

13.1.4 OS Errors

Usually fundamental usage problems, such as Bad Channel Number, Bad Command, etc.

13.2 Error Code Listing

The list of error codes which follows is not necessarily exhaustive, but it does represent all error codes which will normally be returned from DOS XL or any of the Atari device drivers.

Error Code Hex	Decimal	Meaning
$01	1	No error or warning.
$02	2	Truncated ASCII line. The OS did not find a CR within BUFLEN for ASCII line I/O.
$03	3	End of file look ahead. The last byte transfered from the device driver was its end-of-file byte. The device driver must set this status, so it is best to verify that the device being used is capable of returning this status before depending on it.
$80	128	Operation aborted. Set by Device Handler. (Also BREAK abort on Atari.)
$81	129	File already open. Program is trying to open a channel (IOCB) that has already been OPENed.
$82	130	Device does not exist. The device was not found in the OS device table. Often caused by forgetting the disk drive name when using a disk file.
$83	131	File is write only. Program tried to read from a file which can only be used for writing (i.e., file was OPENed with AUX1 set to 8 or 9).
$84	132	Invalid Command. CIO has rejected your requested command. (Example: program tried to do XIO to a device which has no extended operations defined.)
$85	133	Device/File not open. The IOCB has not been OPENed for the operation. Most I/O requests require that the channel

be OPENed before a request can be made.

$86	134	The IOCB specified is invalid. Only IOCB numbers $00, $10, $20, $30, $40, $50, $60, and $70 are valid. From some languages, these will be seen as channels 0 to 7.
$87	135	File is read only. Program tried to write to a file which can only be used for reading (i.e., file was OPENed with AUX1 specified as 4 or 6.
$88	136	End of file. No more data in file.
$89	137	Truncated record error. Usually occurs when the line you are reading is longer then the maximum record size specified in the Call to CIO (line oriented I/O). Can't occur with binary I/O on version 2 OS/A+.
$8A	138	Device timeout error. Usually set by the serial bus I/O handler ("SIO") because a device did not respond within the alloted time as set by the OS.
$8B	139	Device NAK error. Atari: serial I/O error.
$8C	140	Serial framing error. Atari: serial I/O error.
$8D	141	Cursor out of range for specific graphics mode you are in. (Could be used for similar meaning by a non-graphics device.)
$8E	142	Serial bus overflow. Atari: computer could not respond fast enough to serial bus input (SIO error).
$8F	143	Checksum error. Communications over the serial bus are garbled (Atari SIO error).
$90	144	1) Device done error. A valid command on the serial bus was not executed properly. Atari: disk rotational speed needs ad-justment. 2) Write protect error. The diskette has a write protect tab in place.

$91	145	Illegal screen mode error. Bad graphics mode number. Other devices: AUX1 and/or AUX2 bytes in IOCB are illegal.
$92	146	This error means the function you tried to do has not been implemented in the device handler. (Example: attempt to POINT with the graphics device.)
$93	147	Not enough RAM for the graphics mode you requested. (Could be used by custom drivers for a similar message.)

Note: A errors encountered between $A0 through $AD (below) are File Manager errors.

$A0	160	Either a drive # NOT between 1-8 or drive was not powered on.
$A1	161	Too many OPEN files. No free sector buffers to use for another file.
$A2	162	Disk FULL. No free space left on disk.
$A3	163	Fatal system error. Either DOS has bug or bad diskette.
$A4	164	File mismatch. Bad file structure or POINT values wrong.
$A5	165	Bad file name. Check for illegal characters in file name.
$A6	166	The byte count in your POINT Call was greater then 125 (for single density version 2) or 253 (for double density version 2).
$A7	167	The file specified is locked (PROtected). Protected files cannot be erased or written to.
$A8	168	The software interface for the specific device received an invalid command (example: tried to access a non-existent track or sector).

$A9	169	All space allocated for the directory has been used up (too many filenames in use).
$AA	170	The file you requested does not appear on this diskette.
$AB	171	You have tried to POINT to a byte in a file that is not OPENed for update (version 2 only).
$AC	172	Tried to OPEN a DOS 1 file with DOS II (version 2 only).
$AD	173	The disk drive has found bad sectors while trying to format the disk.

Section 14: SHIFTING INTO SYNCHROMESH

14.1 What Should You Already Know Before Reading This?

This discussion assumes you are already familiar with the operation of DOS XL and of your Indus GT disk drive. Many terms will be used in this discussion which are explained in this and other manuals which relate directly with the operation of the Indus GT and DOS XL which are not explained for you in this discussion.

If you are just starting to use DOS XL and your Indus GT, don't start using Synchromesh now! The other manuals which explain the operation of your Indus GT and DOS XL do not expect you to be running Synchromesh, and many things can appear to be completely different when you are using Syncromesh than when you are not.

14.2 What Is "Synchromesh"?

Synchromesh is actually a software program which "extends" the capabilities of your Indus GT for Atari diskette drive. Included with the Synchromesh software program is a new copy of DOS XL (version 2.35 or later) which provides your Atari computer with the necessary capabilities to communicate with an Indus GT which is running in a Synchromesh mode. Synchromesh and the new DOS XL are two completely independent software packages which require each other in order to function to their fullest extent, and so they are supplied by Indus together.

The primary feature of Synchromesh, and one of the primary features of the new DOS XL, is its capability to transfer data between your Atari and your Indus GT drives at speeds which are multiple times faster than ordinary diskette drive data transfers.

The synchromesh software program (called "GTSYNC.COM" on the Synchromesh distribution diskette) contains all the instructions necessary to:

1) Detect which of the drives you have attached to your system are Indus GTs.

2) Send the instructions to your Indus GT drives which are necessary to cause them to start operating in Synchromesh mode.

3) Tell the new version of DOS XL which of the drives attached to your Atari are now operating in Synchromesh mode and which are not.

The new version of DOS XL contains all the instructions necessary to perform the following added features:

1) Communicate with Indus GT drives operating in Synchromesh mode at Synchromesh communication speeds once GTSYNC.COM has told DOS XL which drives are in Synchromesh mode and which are not.

2) Automatically detect the density (either single or double) of a diskette just before any file on the diskette is openned for either reading or writing. This makes the new version of DOS XL "density smart", and prevents you from having to type CONFIG commands in many situations (but not necessary all situations). This feature is strictly part of the new DOS XL, and has no relationship with the Synchromesh feature.

14.3 How Do You Shift Into Synchromesh?

The first step is to boot the Synchromesh distribution diskette in order to start operating under the new copy of DOS XL. Your boot drive (D1:) does not need to be an Indus GT. If DOS XL leaves you in Atari BASIC, type "DOS" to return to DOS XL. If you are using the DOS XL menu program, you will need to "Q"uit to DOS XL. You should have the "D1:" DOS XL command processor prompt at this point. If you have your Atari connected to a speaker (your T.V.'s speaker or a speaker in your video monitor), be sure to turn the volume up enough to hear the loading "beeps". You'll want to compare the speed and tone of these "beeps" to those which are produced once Synchromesh is activated.

To shift into Synchromesh, type:

```
GTSYNC ON
```

The command is just "GTSYNC"; but since this command will both activate and

deactivate Synchromesh, you need to type a SPACE and then the word "ON" to tell GTSYNC you wish to have Synchromesh activated. If you notice that your Indus GT seems to be operating slower than usual while it is loading the "GTSYNC.COM" program, your observation would be correct. This is because your Synchromesh distribution diskette has been specially formatted to operate as fast as possible while running under Synchromesh. Unfortunately, a diskette which is perfectly formatted for operation under Synchromesh is less than perfectly formatted for use while Synchromesh is not active.

Once GTSYNC is loaded and starts running, it will list all eight possible disk drives (D1:-D8:) in order and tell you if any drive responded as the indicated drive number and whether a drive which responded is an Indus GT and is capabable of operating in Synchromesh mode. Once GTSYNC has checked-out all eight possible drives, it then tells the new copy of DOS XL that it is okay to start operating in Synchromesh mode with those drives which are capable of Synchromesh communications.

You can now see the difference between loading a software program using Synchromesh by placing your new Synchromesh distribution diskette into one of the Indus GTs connected to your Atari (in case your number one drive was not a GT), and then typing:

```
D#:GTSYNC ON
```

The "#" in the above command should be replaced with the number of the Indus GT into which you placed the Synchromesh distribution diskette. The "D#:" part is something you should type; it is not the command processor's prompt. If you are listening to the loading "beeps" from your T.V. or monitor speaker, we think you'll be very impressed by their new speed and tone. These "beeps" are directly related to the speed at which data is being transferred between your Indus GT and the Atari.

With Synchromesh active, you are now ready to use the INIT program to start creating a single density backup copy of your Synchromesh distribution diskette. When you use INIT with Synchromesh active to format a diskette on a Synchromesh GT, the diskette will be formatted for the best operation under Synchromesh. If you do not have Synchromesh active, or the drive you use for the format is not a Synchromesh GT, then the diskette will be formatted for the best operation under non-Synchromesh.

Section 14: SHIFTING INTO SYNCHROMESH

Diskettes which are formatted for use under Synchromesh mode are completely compatible, and can be used, under non-Synchromesh mode and in non-Indus GT disk drives (provided the drive used is also capable of reading the density for which the diskette was formatted). Synchromesh mainly influences your GT's communications with the Atari, and is not a new diskette density or anything like that. Synchromesh will work using both single and double density diskettes. Although your GT is capable of reading and writing Atari's seldom used "enhanced" density, Synchromesh cannot be used when a GT is being used with "enhanced" density. This limitation permitted Synchromesh under standard single and the more popular double density (the two standard densities supported by DOS XL) to be used to its maximum potential.

Before you run INIT, you may want to run the CONFIG command to make sure that the Indus GT you will be using to format the new diskette is operating in single density mode. Synchromesh will also function under double density mode, but you should make the backup copy of your Synchromesh distribution diskette in single density since the original distribution diskette is in single density.

When you run the INIT provided with the new DOS XL, you will notice a new menu item. This menu item is a "second step" in the initialization of a Synchromesh formatted diskette. The first step is still to use the first INIT menu item to format the entire diskette. Be sure you are using a Synchromesh GT so that you will get a Synchromesh formatted diskette. Once the drive has formatted the diskette, select the new menu item in order to reformat the boot (or system) tracks of your Synchromesh formatted diskette. This new menu item works almost identically to the first menu item.

The reason for re-formatting the system/boot tracks of Synchromesh formatted diskettes is because Synchromesh is not active while you are first booting your diskette. So, to make booting as fast as possible, the new INIT permits you to re-format just the system tracks of the diskette in the non-Synchromesh format.

PROBLEM PREVENTOR: Do not reformat the system tracks of a diskette which already contains information. Only reformat the system tracks of a diskette which you have just formatted using the first INIT menu item. Some of the information you may have saved onto a diskette may have been placed on part of the system tracks, and reformatting these tracks will destroy this information. Reformatting the system tracks of existing diskettes will only

result in destroying part of the information on these diskettes, and will not result in these diskette automatically running faster.

Once you've both formatted a Synchromesh diskette and then re-formatted the system tracks of the diskette, you should get out of INIT and then run DUPDSK to duplicate the Synchromesh distribution diskette onto your newly formatted diskette. When DUPDSK asks if you want to format the destination diskette, always say "N"o. If DUPDSK reformats the diskette, then you'll loose the special Synchromesh format you created earlier using INIT. If you are listening to your T.V. or monitor speaker, you will notice that DUPDSK will copy the system tracks of your Synchromesh diskette at pretty much the same speed as without Synchromesh active (the "beeps" won't occur as fast even though they are still a higher tone); but once DUPDSK starts reading and writing the non-system tracks, you should notice a sudden increase in speed.

Synchromesh will continue to remain active until you do one of the following:

1) Reboot your Atari.

2) Turn one or more of your Indus GTs off. NOTE: If you turn a Synchromesh GT off and then on again, the GT will no longer be operating in Synchromesh mode; a fact which the DOS XL inside your Atari will not know. This means that DOS XL will not communicate with this GT correctly since DOS XL will try to communicate in Synchromesh mode, and the GT will try to communicate in standard mode.

3) You type the command "GTSYNC OFF" to turn Synchromesh off.

14.4 How Do You Convert Your Existing Diskettes To Synchromesh?

Your existing diskettes will still operate fine with Synchromesh active, except you won't notice any true increase in speed. This is because these diskettes were originally INITialized for non-Synchromesh mode. Due to the way diskette drives in general function, there is no safe way to just "update" your existing diskettes to be faster using Synchromesh. You need to copy all of the information on these slower diskettes onto new diskettes which you have INITialized for use with Synchromesh active.

To do this, use the same process as I described earlier for backing-up your Synchromesh distribution diskette. If you are short on diskettes (or money to buy diskettes), you can first format a new diskette for use with Synchromesh, then copy all the information from a non-Synchromesh diskette onto the new diskette, verify that all the information made it onto the new diskette correctly, re-format the old non-Synchromesh diskette for use with Synchromesh, and then copy the information from the next non-Synchromesh diskette onto this re-formatted diskette. You can continue this process through your entire library.

PROBLEM PREVENTOR: Do not try to update any copy protected or other special diskettes not intended to be used with DOS XL! We will discuss these types of diskettes later in this discussion.

PROBLEM PREVENTOR: The version of DOS.SYS that is written on your new diskette must be 2.35I or higher in order to operate with Synchromesh. If you want your new diskette to autoboot and turn on Synchromesh, you should utilize INIT's Choice 3 (Write DOS.SYS Only) after first Initializing the Disk Only and then Reformatting the Boot Tracks Only (Choices 1 and 4 of INIT).

14.5 How Do You Automatically Activate Synchromesh When Booting?

Even if you activate Synchromesh and then write DOS XL onto a new diskette, DOS XL will not "boot" with Synchromesh active. There are two primary reasons for this:

1) After being powered-on, your Indus GTs must be told to operate in Synchromesh mode before they will respond in that fashion. The reason for this is to insure on-going full compatibility between the Indus GT and many of the protected software diskettes currently being sold; something which is always Indus's number one priority. DOS XL will not boot-up with Synchromesh active since your GTs won't be expecting it.

2) If DOS XL booted-up with Synchromesh active, it would also need to already know which numbered drives are capable of communicating in Synchromesh and which are not. This would require that this information be saved onto the DOS XL diskette along with DOS XL; and if you ever had to change which drives you had connected to your Atari

as certain drive numbers, you'd have to go back and update all of the diskettes on which you have DOS XL saved with this new configuration information.

It is still possible, however, to have Synchromesh automatically activate after DOS XL has booted. Simply place the command "GTSYNC ON" into a "STARTUP.EXC" file on your system diskette. Be sure to also include the "GTSYNC.COM" utility on your system diskette. The STARTUP.EXC file is discussed in the DOS XL Reference Manual.

14.6 What About The New DOS XL's Auto-Density Feature?

In addition to the new DOS XL's (version 2.35 or later) ability to communicate with the higher speed Synchromesh GTs, it also has the ability to auto-determine when a diskette is single density or double density. DOS XL always checks a special location on a special sector on track zero of any diskette just before trying to open a file for reading or writing on that diskette. This location tells DOS XL the density of the diskette containing the file. Most diskette operations (although not all) start by either your program or some other pre-written program trying to open a file for either reading or writing, and once this occurs DOS XL will automatically adjust for any diskettes which have been changed and are of a different density from the previous diskette.

This special piece of density information is located on a sector on the diskette which is always treated by DOS and your drive as single density, regardless of the actual density of the diskette.

Since DOS XL requires more memory space to handle double density diskettes than single density diskettes, DOS XL now assumes that all diskettes will be double density when it is first booted. This means that DOS XL will always allocate enough memory to handle double density diskettes, and therefore it may use up more of your Atari's memory than did your previous copies of DOS XL if you normally used one or more of your drives in single density. The new DOS XL will not take up much more memory than that for which you are accustomed if you normally use double density diskettes. But remember, DOS XL contains new added features without getting rid of any old features; so it does require a little more memory for these new features.

Ordinarily, DOS XL's auto-density feature will handle any diskette changes you make without any problem. But, there are certain circumstances in which you will need to assist DOS XL's decision concerning a diskette's density. That is why the CONFIG command is still available for use with the new DOS XL. Some of the more typical circumstances are as follows:

1) When your drive is currently operating in a density different than the density for which you wish to format a diskette. This circumstance is only valid if you have two or more drives, since single drive users would still use INITDBL. The way to handle this situation is to first insert the diskette you wish to format into the drive, and then use the CONFIG command to CONFIGure the drive to the density in which you wish to format the diskette. After CONFIG, use INIT to initialize the diskette. Remember: Neither CONFIG nor INIT try to open any files on any diskettes (although the command processor will open CONFIG.COM and INIT.COM on your default drive when you issue the CONFIG and INIT commands), so your non-default drives will remain in the density for which you CONFIGured them until you've started the diskette INITializing.

2) When you wish to re-format a diskette using a density different from that for which the diskette is already formatted. This situation is actually no different than the first situation, since neither CONFIG nor INIT will try to open any files on the diskette.

14.7 What About Synchromesh And Software Other Than DOS XL?

Although your Indus GT is capable of communicating with your Atari at a much higher speed than most other diskette drives, it is normally prevented from doing this because the Atari does not expect a diskette drive to be able to communicate at this higher speed. This causes a problem for Synchromesh, since it needs "something" to tell your Atari to communicate with your GT at a faster rate before your GT can start communicating at this faster rate. And, even if Synchromesh communication speeds were to "magically" work without any "something" to help your Atari, the higher communication rates would interfere with many software protection schemes and would cause incompatibility between your GT and some of the programs you may have purchased. And as we mentioned

before, compatibility is always the first priority at Indus.

So, special non-DOS XL programs will not just "magically" take advantage of your GT's Synchromesh capabilities. What can you do about this? Always check with your software dealer(s) to see if the software package for which you are interested is available in a newer "Indus GT Synchromesh enhanced" version. Normally these newer software packages contain special markings to indicate they have been upgraded. And, if you have existing software packages which you'd like to take advantage of Synchromesh, contact the software's publisher. It is always possible that the publisher of your favorite package may have an upgrade policy which may permit you to get a new "Indus GT Synchromesh enhanced" version of the software for little or no charge, especially if you remembered to send in the publisher's software registration card.

If, for some reason, your favorite nationally advertised software publisher has not yet upgraded their software packages to perform at Synchromesh speeds, have them contact our support staff here at Indus (818/882-9600) if they haven't already done so; and we'll be happy to supply them with all the necessary information free of charge.

APPENDIX A

A SHORT GLOSSARY OF COMPUTER TERMINOLOGY

ABSOLUTE ADDRESS The address or location in memory that represents the actual, physical location in memory for that particular piece of information. This contrasts with relative or indirect addresses -- see below.

ACCESS TIME This is the amount of time it takes for information to become available once the computer has requested it. For RAM memory, this time is measured in billionths of a second. For disk drives, the access time is measured in thousandths of a second and includes the time required to move the read/write head over the requested track, as well as the latency or time required for the correct sector to appear under the read/write head.

ACRONYM A word formed from the first letter or other letters of a compound or lengthy expression. For instance, PTA represents Parent Teacher Association or RAM represents Random Access Memory. Acronyms are usually printed in all capital letters.

ADDRESS Refers to a specific location in memory. With 8 bit microprocessors the range of these addresses is from 0 to 65536 decimal or 0 to FFFF in hexidecimal notation. A memory map gives you the location or address where specific ROM or RAM routines used by your computer may be found.

ALPHANUMERIC A character set containing any combination of letters (a-z), numbers (0-9) or miscellaneous characters (#$@!%&* etc.).

ANSI An acronym for American National Standards Institute, a committee formed to establish standards for data processing and computer communications, recording media, hardware and software interface specifications, etc.

APPLICATION A program which is designed for a specific purpose, for example, a real estate investment package or a home accounting system.

ARGUMENT The number, variable or string which follows and gives definition to a function or command.

APPENDIX A

ASCII An acronym for American Standard Code for Information Interchange -- a set of standard numerical equivalents for the alphabet, numbers, punctuation and special symbols used by the computer.

ASSEMBLER A computer program which converts assembly language instructions (easier for the programmer to remember) into machine language instructions (easier for your computer to perform).

ASSEMBLY LANGUAGE A programming language that is more advanced than machine language. Assembly language uses easier to remember symbols or mnemonics (for example, LDY means LoaD Y or load the Y register of the microprocessor, etc.) which have one for one equivalents in machine language.

BACKUP Making an extra copy of a master diskette (or a data diskette) or periodically saving your files to diskette while work is in progress so that in the event of disaster (computer malfunction, power outage, spilling coffee on the diskette) files are not lost.

BASIC Beginner's All-purpose Symbolic Instruction Code, a programming language that is commonly used in many microcomputers.

BATCH A set of commands or data that has been grouped for processing sequentially in a single session.

BAUD A unit of measurement for signalling speed, normally bits per second, as a measure of the flow of data.

BCD Binary Coded Decimal. A method of using groups of four binary digits to represent each individual digit of a decimal number.

BINARY A numbering system used by all computer systems which is based on two digits (0,1).

BIT An acronym for Binary digIT (0 or 1). There are eight bits in a byte.

BLOCK A term representing a contiguous group of data treated as a single unit.

BOOT A term used to describe loading a computer program or operating system. A "cold boot" is starting with the machine off. A "warm boot" is loading the program or operating system without turning off the computer.

BUFFER A storage area used to hold data temporarily or to compensate for the rate of data flow.

BUG An error, flaw or malfunction in either the software (program) or hardware (computer and/or peripherals).

BYTE (1) A series of binary digits (usually eight bits) which together represent a single character, or (2) a unit of computer storage.

CARRIAGE RETURN (sometimes shortened to RETURN) The term was derived from the typewriter lever used to advance to the beginning of the next line. In computer terminology it represent the non-printed character that returns the cursor to the beginning of the next line on a CRT or a print head to the beginning of the line (with or without a line feed) on a printer.

CATALOG Same as a directory; a list of items such as files or programs which are arranged in an easy to reference structure.

CATHODE RAY TUBE (CRT) A device similar to a television tube, but used for displaying computer data and graphics.

CHANNEL A communication link through which data and/or control information can be transmitted.

CHARACTER A letter, a number or a miscellaneous symbol such as !@$%&*?.

CHECKSUM A number which represents the sum of a series of characters or numbers. Typically such a number appears at the end of a sector or block of data and is used to verify the accuracy of the data which was read -- if the checksum number calculated as the sector is being read matches that which was previously recorded, it is much more likely that the sector was correctly read.

CHIP A small piece of silicon onto which an integrated, microscopic circuit is built. These tiny objects have revolutionized the computer world.

APPENDIX A

CLOCK A highly accurate pulse or signal that is used by your computer to derive timing information or synchronize the transfer of data or control information.

CODE A term used to describe computer instructions or symbols describing data.

COMMAND The portion of an instruction for the computer that specifies the operation which is to be performed.

COMPILER A program that translates high level program code into machine language code.

COMPUTER LANGUAGE A set of coded instructions which are used for communicating with the computer.

CONCATENATION Tying two strings together. In BASIC this is done by using a "+" sign, i.e. NF$+NL$.

CONTROL KEY (often abbreviated as CTRL) Usually used in conjunction with another key pressed at the same time to indicate a special cursor or program function.

CPS Characters Per Second.

CPU Central Processing Unit. The central processor or "brains" of a computer containing main storage, arithmetic unit, and registers.

CRC Cyclic Redundancy Check. A more sophisticated and comprehensive mathematic operation performed on data to ensure better transmission accuracy than a simple CHECKSUM operation.

CRYSTAL A component (quartz crystal) commonly found in computer systems that due to its precise molecular structure is used to generate highly accurate clock signals or timing pulses.

CURSOR A small mobile square or dash (sometimes flashing) which runs along the screen indicating where the next character will appear.

DAISYWHEEL A type of print element that is circular or wheel shaped, with daisy like petals on which different printable characters are formed. Daisywheel printers produce excellent quality, fully formed characters, and are often referred to as typewriter or letter quality printers.

DATA A general term which describes raw information such as numbers, facts, symbols, characters, names, information, etc.

DEBUG The process of finding and fixing errors in a computer program.

DECIMAL The standard, base 10 counting system which utilizes the digits 0 through 9.

DEFAULT VALUE The path the computer program will take unless other options are specifically indicated by the user.

DENSITY This refers to how closely data is packed on a disk. Track density refers to how closely the concentric circles or tracks of information are spaced. Single and Double Density refers to how closely individual bits are packed around each circle or track. Double density (MFM) encoding places twice as much data in the same linear space on the track as single density (FM) encoding.

DIRECTORY A list or catalog of all of the files contained on a diskette or other storage medium.

DISK DRIVE A peripheral device that reads and writes data to magnetic disks.

DISK FILE A file that resides on a disk.

DISK OPERATING SYSTEM (DOS) The software or programs that enable the computer to make use of disk drives for storing and retrieving information. The disk operating system also typically takes care of checking CRC bytes for read/write errors, allocation of space on the diskette, specific file or data location on the diskette, formatting of blank diskettes, and so on.

DOCUMENTATION Instructions accompanying computer programs explaining exactly how these programs work.

APPENDIX A

DOT MATRIX PRINTERS A printer device which prints characters from a matrix of dots (often 5 x 7) which strike the paper in varying configurations depending on the desired character symbol.

DOWN LOAD Transmitting information from one computer system to another.

DOWN TIME The time when a computer is not functioning due to mechanical, electronic or human failure.

DRIVER This refers to small programs that are used to run other programs or to control external devices or peripherals.

ESCAPE (ESC) A non-printed character that is typically interpreted by the computer or printer (in conjunction with another character) as a control signal to perform a specific function.

EXECUTE This term refers to the process of first interpreting the instruction in question and then performing the requested operation(s).

EXTENSION This refers to a supplemental set of characters often appended to a file name (and usually seperated by a period); extensions are typically used to categorize file types (e.g., .BAS might be appended to BASIC file names while a file labeled .TXT might be appended to letters or text files).

EXTRINSIC COMMAND (also referred to as TRANSIENT COMMAND FILES) Refers to commands that are not located within the computer's memory and must be looked up and loaded from a disk before they can be performed.

FILE A group of organized sectors or data, that is assembled for a single function and is considered a single unit; this is used similar to the way a file folder would hold a set of related information in a filing cabinet.

FIRMWARE Logic circuits in computer ROM which can be altered by software under certain circumstances. Compare "hardware" and "software".

FLOPPY DISKETTE A flat, flexible disk of magnetic media 8, 5-1/4 or 3-1/2 inches in diameter used to store data.

FORMAT (1) The process of placing tracks and sectors onto a diskette prior to use in the computer. This process destroys all existing data on the diskette, or (2) the arrangement of text, punctuation, indents, etc. on reports or a word processor.

GIGO Garbage In Garbage Out. The input of unwanted, meaningless information results in the output of unwanted, meaningless information.

HANDSHAKING Signals between the computer and various peripherals that confirm the sending or receiving of data.

HARDWARE The computer and all its electronic and physical parts -- the keyboard, printer, monitor, disk drive, etc.

HARD ERROR An error that consistantly occurs with repeated re-trys. In most systems, a hard error condition is not brought to the attention of the user unless it has occurred on at least 10 (and in some systems many more) successive retrys.

HEAD This refers to the magnetic read/write head which converts the computer's binary signals to magnetic pulses on the diskette.

HERTZ (HZ) A unit of measurement representing cycles per second. In the United States AC power is nominally rated at 115 volts, 60 hertz.

HEXIDECIMAL A base 16 numbering system used by the computer. The digits 0 - 9 and A, B, C, D, E and F are used for notation. Hexidecimal numbers are normally preceded by a dollar sign symbol ($) to avoid confusing them with decimal or octal numbers.

HIGH LEVEL LANGUAGE This refers to computer languages which utilize instructions that are more english like and therefore simpler for the programmer to use and understand; each high level language command usually translates into a series of simpler machine language statements.

HI-RES GRAPHICS HIgh RESolution Graphics. A method for creating computer graphics which allows the use of a single point on the screen, which in turn produces finer detail than lo-res or keyboard graphics.

HOME A command from a key or a program which causes the cursor to move to its "home" position (usually the upper left-hand corner of the screen).

INDIRECT ADDRESS Rather than giving the specific location in memory for a particular piece of information, INDIRECT ADDRESSing gives instead a place in memory where the specific location of this information (or absolute address) may be found.

INITIALIZE (1) Setting various parameters to an initial value prior to processing, or (2) the process of placing tracks and sectors onto a diskette prior to use in the computer. This process destroys all existing data on the diskette.

INPUT Information or data to be entered into computer storage or usage by various methods.

I/O Input/Output. Refers to the flow of information both into and out from the computer or peripheral.

INTERFACE An electronic link which connects two computer components making them compatible.

INTRINSIC COMMAND Refers to a command that can be performed directly by the computer from memory without needing to load portions of the command from disk.

INVERSE VIDEO Screen output which is written as dark characters on a light background.

JUSTIFICATION In word processing this represents the process of controlling the spacing of words and/or graphics so that an even, vertically aligned right or left hand margin is formed.

KEYBOARD GRAPHICS Drawing with defined characters such as triangles, circles and bars that substitute for alphanumeric characters.

KBYTE (K) A unit of measure of computer memory. 1 Kbyte = 1024 bytes and is sometimes called a kilobyte.

LEAST SIGNIFICANT BIT (or BYTE) The bit or byte in the rightmost position in a number or word. For instance in the number 12345, the number 5 is the least significant or lowest value number, since the 4 represents 40, the 3 represents 300, and so on.

LED Light Emitting Diode. A relatively low power, highly reliable source of light. Light Emitting Diodes are frequently used as indicators or status displays on computers and are also used in disk drives (coupled with photo-detectors) to detect track zero positioning of the read/write head or the presence or absence of a write protect tab on your diskette, etc.

LOMEM An acronym for LOw MEMory, a specific location in memory that points to the lowest address in RAM that a user program can be loaded.

LO-RES GRAPHICS LOw RESolution Graphics. A method for creating computer graphics using small squares of color usually the size of a single character. This creates a stepped appearance in circles and diagonal lines. Uses less memory than hi-res graphics.

MACHINE LANGUAGE The lowest level computer language; directly understandable by the computer.

MAINFRAME (1) The central processing unit of a very large computer, or (2) the first generation of computers, which are huge by today's standards.

MEGABYTE (MB) A unit of measure of computer memory. 1 MB = 1,024,000 bytes.

MEMBRANE KEYBOARD A keyboard with a flat, flexible keyboard surface. Used on cash registers in some fast food resturants.

MEMORY The amount of electronic workspace available on a computer. Usually measured in Kbytes.

MENU A list of programs, options, procedures or other information which is displayed on the screen and serves as an aid to the user in selecting a task.

MENU-DRIVEN A feature of computer programs using menus to aid the user, alleviating the need to refer constantly to documentation.

APPENDIX A

MICROCOMPUTER A general term which refers to a complete small desk-top computer system.

MICROPROCESSOR A silicon chip that is the central processing unit (CPU) of a microcomputer.

MINI COMPUTER A mid-range computer system, smaller than a mainframe but larger than a microcomputer.

MNEMONIC An aid to memory, e.g., in naming a program, variable command, etc.

MODEM MOdulator/DEModulator. A device which converts computer signals to codes which can be transmitted via telephone lines to another computer.

MOST SIGNIFICANT BIT (or BYTE) The bit or byte in the leftmost position in a number or word. For instance in the number 12345, the 1 is the most significant or highest value element since it represents 10,000, whereas the 2 represents only 2,000, the 3 represents 300, and so on.

MTBF Mean Time Between Failures. The average time between failures for any device.

MTTR Mean Time To Repair. An average time required by a technician to repair the faulty device.

NIBBLE Four bits or half a byte. Usually these represent the four least significant bits or most significant bits of a byte. Nibbles are often used to create BCD numbers (binary coded decimal).

OEM Original Equipment Manufacturer.

OPERATING SYSTEM (OS) The software or firmware control system under which all other software and hardware in the computer functions. The operating system manages all input/output, peripherals and deals with the application software for the computer's main processor.

OUTPUT Information or data which is transmitted from computer storage to the screen, printer or other peripheral device.

OVERFLOW In communications, a condition in which the buffer cannot hold all of the data being sent to it. Can also be a condition caused by a mathematical operation that generates a number that is too large for the software or hardware of the particular machine.

PAGE 256 consecutive bytes of memory beginning on a page boundary. Page boundaries begin at address 0 and occur each 256 bytes thereafter (multiples of $100).

PARAMETER Values which follow a command or instruction that provide additional information for the command.

PARALLEL Having the same direction or occurring simultaneously. Usually refers to the transmission of information eight or more bits at a time, rather than serially or only one bit at a time.

PARITY CHECK The simplest of the data transmission error detecting techniques. A bit (usually bit 8) is set to 1 or 0 to indicate whether the remaining seven bits in the byte contain an odd or even number of "1" bits.

PEEK A BASIC programming statement used to read or display the decimal value of a specified location in memory.

PERIPHERAL Any device for input/output in a computer system such as a printer, monitor, plotter, disk drive, etc.

PIXEL The basic element (dot) of a monitor screen which is definable and addressable and commonly used for computer graphics.

PLOTTER A peripheral device used to draw graphs, plot coordinate points, etc.

POINTER A location in memory which contains the location of another specific item of information.

POKE A BASIC programming statement used to place a decimal number directly into a specified location in memory.

PRINTED CIRCUIT BOARD (PCB or PC Board) A fiberglass board onto which a copper wiring diagram has been etched -- this copper plated wiring electrically connects each of the components that are later installed on the board.

PROGRAM A sequence or series of instructions given to the computer to perform a specific series of tasks or functions; also referred to as "software".

PROMPT Direction given by the computer to the user. For example, many computer programs offer prompt lines at the bottom of the screen reminding the user what commands are available in that particular mode.

PROTECT A method of securing a file or data against inadvertant or unintentional modification. This can be accomplished through physical protection of the diskette file or by software recording special indicators on the directory entry of the file.

PSEUDO-RANDOM A sequence of numbers generated by the computer by a known formula, yet done in such a way that it is virtually impossible for the user to predict what number will be chosen. Used primarily in video games.

RAM Random Access Memory. The computer's workspace. Information stored in RAM is lost when the computer is turned off unless the data there is saved to disk or tape before exiting.

RANDOM ACCESS This refers to the ability of the computer or disk drive to go directly to the sector or segment of data it wishes to use without the necessity of sequentially searching through an entire record to reach the desired information.

REGISTER A register represents a temporary storage or processing area internal to the microprocessor which is accessable much more quickly than main memory.

RELATIVE ADDRESS Rather than giving a specific location in memory where a certain piece of information can be found, RELATIVE ADDRESSing tells you how many bytes higher or lower in memory than currently located this information can be found.

REMARK A comment that can be made in a computer program by the programmer to serve as a reminder, e.g., "REM the next 12 lines are a subroutine that plays music".

RESERVED WORDS A list of words and abbreviations that BASIC recognizes as commands or functions. The reserved words cannot be used in variable names.

RESET This term describes a software or hardware command which can be sent to your computer that will typically 1. Halt any program currently in operation. 2. Initialize or restore to power on condition defaults a number of variables. 3. Will not normally erase any BASIC program currently in memory (in this respect it has advantages over turning power to the computer off and on again).

ROM Read Only Memory. Permanent computer memory containing such information as the operating system, character set, etc. This information is not alterable and is not lost when the computer is turned off as with RAM.

ROUTINE A set of instructions or commands organized to direct the computer to perform specific action(s). A program can contain several routines although a routine may be considered a program by itself.

RS232 or **RS232C** This represents a widely used serial interface standard (adopted by the Electronic Industry Association) that governs the timing, control signals and physical connectors used between computers and various peripherals.

SAVE TO DISK Giving the computer a command to permanently save what is in RAM memory on a disk so that the computer can be turned off or a new program loaded.

SERIAL Handling data sequentially or one bit at a time in the transfer of data.

SCROLLING Movement of the text up or down so that material which is not visible in one frame can be displayed on the screen.

SECTOR A sector represents a block of data (typically 128 or 256 bytes in length) which constitutes one of several segments or units composing a track.

SEEK TIME The amount of time required for the disk drive's positioning system to move the read/write head to the specified track.

SOFT ERROR A transient, not predictable fault encountered by the disk drive. This can be caused by dust particles, sudden vibration of the disk drive or media, power line surges, or other causes. Soft errors, by definition, are recovered from by re-reading the sector or block of data originally producing the error.

SOFT-SECTORED This refers to diskettes on which the sectors or blocks of information comprising each track are identified by address data being recorded magnetically on the track by software. This contrasts with hard sectored diskettes where the sector separations are marked by reference holes being physically punctured through the diskette.

SOFTWARE Computer programs and sub-routines. In contrast, the computer and its various peripherals are hardware.

STEPPER MOTOR A motor that rotates in precise increments each time an electrical pulse is applied to it. In flexible disk drives, these movements position the read/write head over the concentric tracks on the diskette where data is stored.

STRING A related set of alphanumeric characters or symbols. The word "INDUS" is a string of five characters. String variables (as opposed to real number or integer variables) are usually indicated with a dollar sign suffix; for example, X$ = DATA.

SUBROUTINE A self-contained program within a program. When a portion of a program has to be executed several times it can be incorporated into a subroutine and then accessed whenever necessary.

SYNCHRONOUS DATA TRANSMISSION The process of sending and receiving information having both the sending and receiving devices synchronized to a common clock signal.

SYNTAX The grammatical rules of a programming language or operating system.

TIME SHARING A technique whereby many terminals can share a centralized computer at the same time.

TRACK Information on a diskette is organized in the form of concentric circles or tracks of recording area. Each of these tracks can be randomly accessed by the disk drive by moving its read/write head over any selected track location and does not require reading all of track one before moving to track two, etc.

TRACKS PER INCH (TPI) A method of measuring the density of concentric circles or tracks of data on disks. On minifloppy diskettes, 48 tracks per inch is standard track density, 96 tracks per inch is referred to as double track density.

TRANSMISSION SPEED This is the speed at which data is sent from one device, such as the computer, to another device, such as a printer or modem. Transmission speed is usually measured in the number of bits per second or BAUD rate.

TRUNCATE To shorten by cutting off.

USER-FRIENDLY A term used by the computer industry to describe a program that is easy to use without extensive documentation.

USER MEMORY Memory that can be accessed and changed by the user. This is usually the portion of the computer's addressing range that remains once the operating system and BASIC ROM areas have been removed.

UTILITY Non-application programs that allow the user to perform various functions closer to machine level than normal. Their varied purposes include copying files, renumbering programs, compiling code to make the program run faster, etc.

VARIABLE A symbol used in a program to which a value can be assigned. Variables can represent real numbers (e.g., LET X = 10) or, when followed by a "$" symbol or string indicator, can represent a group or string of characters or numbers (e.g., LET G$ = BURLESQUE).

WILD CARD Symbols used to represent one or more characters of any value. The expression is derived from and is used in a similar way to card games, where the "wild card" can be used to represent any card chosen by the dealer or in this case the computer.

WORD PROCESSOR A computer program which allows the computer to function as a powerful typewriter. Text files can be manipulated quickly and easily -- blocks of text can be moved without retyping, spelling errors corrected, editorial or global changes made, etc.

WRITE ENABLE The process of modifying storage media to allow a disk or tape drive to record information on this media. On a minifloppy diskette, this involves removing the write protect sticker or tab from the media, or alternately on an Indus GT, by pressing the protect button to turn off the electronic write protect function.

WRITE PROTECT The process of physically inhibiting your computer or disk drive from being able to intentionally or accidently record information on your diskette. This is accomplished on a minifloppy diskette by placing a write protect tab or sticker over the write enable notch on the side of the diskette jacket, or alternately on an Indus GT, by pressing the protect button to turn on the electronic write protect function.

APPENDIX B

DECIMAL/HEXIDECIMAL/BINARY CONVERSION TABLE

Decimal	Hex	Binary	Decimal	Hex	Binary
0	0	00000000	32	20	00100000
1	1	00000001	33	21	00100001
2	2	00000010	34	22	00100010
3	3	00000011	35	23	00100011
4	4	00000100	36	24	00100100
5	5	00000101	37	25	00100101
6	6	00000110	38	26	00100110
7	7	00000111	39	27	00100111
8	8	00001000	40	28	00101000
9	9	00001001	41	29	00101001
10	A	00001010	42	2A	00101010
11	B	00001011	43	2B	00101011
12	C	00001100	44	2C	00101100
13	D	00001101	45	2D	00101101
14	E	00001110	46	2E	00101110
15	F	00001111	47	2F	00101111
16	10	00010000	48	30	00110000
17	11	00010001	49	31	00110001
18	12	00010010	50	32	00110010
19	13	00010011	51	33	00110011
20	14	00010100	52	34	00110100
21	15	00010101	53	35	00110101
22	16	00010110	54	36	00110110
23	17	00010111	55	37	00110111
24	18	00011000	56	38	00111000
25	19	00011001	57	39	00111001
26	1A	00011010	58	3A	00111010
27	1B	00011011	59	3B	00111011
28	1C	00011100	60	3C	00111100
29	1D	00011101	61	3D	00111101
30	1E	00011110	62	3E	00111110
31	1F	00011111	63	3F	00111111

APPENDIX B

Decimal	Hex	Binary	Decimal	Hex	Binary
64	40	01000000	96	60	01100000
65	41	01000001	97	61	01100001
66	42	01000010	98	62	01100010
67	43	01000011	99	63	01100011
68	44	01000100	100	64	01100100
69	45	01000101	101	65	01100101
70	46	01000110	102	66	01100110
71	47	01000111	103	67	01100111
72	48	01001000	104	68	01101000
73	49	01001001	105	69	01101001
74	4A	01001010	106	6A	01101010
75	4B	01001011	107	6B	01101011
76	4C	01001100	108	6C	01101100
77	4D	01001101	109	6D	01101101
78	4E	01001110	110	6E	01101110
79	4F	01001111	111	6F	01101111
80	50	01010000	112	70	01110000
81	51	01010001	113	71	01110001
82	52	01010010	114	72	01110010
83	53	01010011	115	73	01110011
84	54	01010100	116	74	01110100
85	55	01010101	117	75	01110101
86	56	01010110	118	76	01110110
87	57	01010111	119	77	01110111
88	58	01011000	120	78	01111000
89	59	01011001	121	79	01111001
90	5A	01011010	122	7A	01111010
91	5B	01011011	123	7B	01111011
92	5C	01011100	124	7C	01111100
93	5D	01011101	125	7D	01111101
94	5E	01011110	126	7E	01111110
95	5F	01011111	127	7F	01111111

Decimal	Hex	Binary	Decimal	Hex	Binary
128	80	10000000	160	A0	10100000
129	81	10000001	161	A1	10100001
130	82	10000010	162	A2	10100010
131	83	10000011	163	A3	10100011
132	84	10000100	164	A4	10100100
133	85	10000101	165	A5	10100101
134	86	10000110	166	A6	10100110
135	87	10000111	167	A7	10100111
136	88	10001000	168	A8	10101000
137	89	10001001	169	A9	10101001
138	8A	10001010	170	AA	10101010
139	8B	10001011	171	AB	10101011
140	8C	10001100	172	AC	10101100
141	8D	10001101	173	AD	10101101
142	8E	10001110	174	AE	10101110
143	8F	10001111	175	AF	10101111
144	90	10010000	176	B0	10110000
145	91	10010001	177	B1	10110001
146	92	10010010	178	B2	10110010
147	93	10010011	179	B3	10110011
148	94	10010100	180	B4	10110100
149	95	10010101	181	B5	10110101
150	96	10010110	182	B6	10110110
151	97	10010111	183	B7	10110111
152	98	10011000	184	B8	10111000
153	99	10011001	185	B9	10111001
154	9A	10011010	186	BA	10111010
155	9B	10011011	187	BB	10111011
156	9C	10011100	188	BC	10111100
157	9D	10011101	189	BD	10111101
158	9E	10011110	190	BE	10111110
159	9F	10011111	191	BF	10111111

APPENDIX B

Decimal	Hex	Binary	Decimal	Hex	Binary
192	C0	11000000	224	E0	11100000
193	C1	11000001	225	E1	11100001
194	C2	11000010	226	E2	11100010
195	C3	11000011	227	E3	11100011
196	C4	11000100	228	E4	11100100
197	C5	11000101	229	E5	11100101
198	C6	11000110	230	E6	11100110
199	C7	11000111	231	E7	11100111
200	C8	11001000	232	E8	11101000
201	C9	11001001	233	E9	11101001
202	CA	11001010	234	EA	11101010
203	CB	11001011	235	EB	11101011
204	CC	11001100	236	EC	11101100
205	CD	11001101	237	ED	11101101
206	CE	11001110	238	EE	11101110
207	CF	11001111	239	EF	11101111
208	D0	11010000	240	F0	11110000
209	D1	11010001	241	F1	11110001
210	D2	11010010	242	F2	11110010
211	D3	11010011	243	F3	11110011
212	D4	11010100	244	F4	11110100
213	D5	11010101	245	F5	11110101
214	D6	11010110	246	F6	11110110
215	D7	11010111	247	F7	11110111
216	D8	11011000	248	F8	11111000
217	D9	11011001	249	F9	11111001
218	DA	11011010	250	FA	11111010
219	DB	11011011	251	FB	11111011
220	DC	11011100	252	FC	11111100
221	DD	11011101	253	FD	11111101
222	DE	11011110	254	FE	11111110
223	DF	11011111	255	FF	11111111

APPENDIX C

OTHER FILES ON YOUR DISKETTE

C.1 CONFIG.BAS A Guide To Writing Your Own Density Utility

The following program listing is offered as a guide or model for the advanced programmer who may wish to write his own density configuration program or incorporate such a routine within some other program he wishes to write. As we have previously noted, special, highly technical information of this type will receive only modest levels of support from our technical support staff.

```
 10 REM BASIC DISK DRIVE STATUS/CONFIGURATION PROGRAM
20 REM BY GREG FARRIS   2/13/84
30 DIM TBL$(12),EXC$(4),DEN$(7),DP$(3)
40 TBL$="              "
50 OPEN #1,4,0,"K:"
60 GRAPHICS 0:POKE 752,1
70 POKE 53774,112:POKE 16,112:? :? "WHICH DRIVE (1-8)":? "OR ESC TO EXIT?":
GET #1, DRIVE
80 IF DRIVE=155 THEN 70
90 IF DRIVE=27 THEN POKE 580,0:RESTORE 1000:FOR X=1 TO 4:READ E:EXC$(X, X)=CHR$(E):
NEXT X:X=USR(ADR(EXC$))
100 IF DRIVE<49 OR DRIVE>56 THEN 70:? DRIVE-48
110 RAMTOP=HI=PEEK(RAMTOP)-20
120 RAM=HI*256-1:POKE 769,DRIVE-48
130 DCB=768:REM BASE ADDRESS OF DEVICE CONTROL BLOCK
140 GOSUB 470:GOSUB 740
143 REM
145 REM *** SELECT SIMPLE OR CUSTOM CONFIGURATION ***
150 ? :? :? "SIMPLE OR CUSTOM CONFIGURATION?";GET #1,X:IF X=155 THEN 150
160 IF X=83 THEN GOSUB 280:GOTO 470
170 IF X=67 THEN GOSUB 580:GOTO 190
180 IF X<>67 OR X<>83 THEN 150
183 REM
185 REM *** WRITE TO OPTION TABLE ***
190 POKE DCB,49:REM SERIAL BUS I. D.
200 POKE DCB+1,DRIVE:REM DRIVE NUMBER
210 POKE DCB+2,79:REM WRITE TO OPTION TABLE
220 POKE DCB+3,128:REM STATUS AS OUTPUT
230 POKE DCB+5,HI:POKE DCB+4,0:REM BUFFER ADDRESS HIGH AND LOW
240 POKE DCB+6,15:REM TIMEOUT VALUE
250 POKE DCB+8,12:REM NUMBER OF BYTES
260 POKE DCB+9,0:REM HIGH BYTE NUMBER OF BYTES
270 GOSUB 530:POKE 580,0:RESTORE 1000:FOR X=1 TO 4:READ E:EXC$(X,X)=CHR$(E):
NEXT X:X=USR(ADR(EXC$))
```

```
273 REM
275 REM *** SIMPLE CONFIGURE ***
280 ? :? :? ;" WHICH DENSITY:":? :? ;" 1. SINGLE":? ;" 2. DOUBLE":
? ;" 3. STATUS ONLY"
290 GET #1,DEN
300 IF DEN=155 THEN DEN=50:GOTO 360
310 IF DEN<49 OR DEN>51 THEN 290
320 IF DEN=51 THEN GOSUB 740:GOTO 70
330 IF DEN=49 THEN GOSUB 420:GOTO 190
340 IF DEN=155 THEN DEN=50
345 REM
350 REM *** DOUBLE DENSITY ROUTINE **
360 FOR X=1 TO 10
370 TBL$(6,6)=CHR$(4):TBL$(7,7)=CHR$(1):TBL$(8,8)=CHR$(0)
380 TBL=ASC(TBL$(X,X))
390 POKE RAM+X,TBL
400 NEXT X:GOSUB 190
405 REM
410 REM *** SINGLE DENSITY ROUTINE **
420 FOR X=1 TO 10
430 TBL$(6,6)=CHR$(0):TBL$(7,7)=CHR$(0):TBL$(8,8)=CHR$(128)
440 TBL=ASC(TBL$(X,X)):POKE RAM+X, TBL
450 NEXT X:RETURN
455 REM
460 REM *** READ OPTION TABLE ***
470 POKE DCB+2,78:REM READ OPTION TABLE
480 POKE DCB+3,64:REM STATUS AS INPUT
490 POKE DCB+4,0:REM BUFF ADDR LOW
500 POKE DCB+5,HI:REM BUFF ADDR HIGH
510 POKE DCB+6,7:REM TIMEOUT VALUE
520 POKE DCB+8,12:REM NUMBER OF BYTES
530 RESTORE :FOR X=1 TO 4:READ D:EXC$(X,X)=CHR$(D):NEXT X
540 X=USR(ADR(EXC$)):REM EXECUTE OPTION TABLE COMMAND
550 IF PEEK(DCB+3)=138 THEN ? "DRIVE ";DRIVE-48;" DOES NOT RESPOND!":GOTO 70
560 IF PEEK(DCB+3)=139 THEN ? "DRIVE ";DRIVE-48;" NOT CONFIGURABLE!":GOTO 70
570 RETURN
573 REM
575 REM *** CUSTOM CONFIGURE ***
580 ? " TYPE IN DESIRED VALUE":? "OR RETURN TO LEAVE UNCHANGED":POKE 752,0
590 TRAP 610
600 ? :? :? "TRACKS=";TR;:? "";:INPUT TR:POKE RAM+1,TR
610 TRAP 630
620 ? "STEP RATE IN MSECS. ";SR;:? "";:INPUT TR:POKE RAM+2,TR
630 TRAP 650
640 ? "SECTORS PER TRACK=";ST;:? "";:INPUT ST:POKE RAM+4,ST:POKE RAM+3,0
```

```
650 ? "NUMBER OF SIDES?";SIDES;:? "";:GET #1,SIDES:IF SIDES>50 THEN ? :
GOTO 670
660 ? SIDES-48:POKE RAM+5,SIDES-49
670 ? "DENSITY?";DEN$;
680 ? "";:GET #1:IF X=68 THEN PRINT "DOUBLE":POKE RAM+6,4:GOTO 700
685 IF X=155 THEN ? :GOTO 700
690 ? "SINGLE":POKE RAM+6,0:DEN$="SINGLE"
700 ? "BYTES/SECTOR=";BS;:? "";:INPUT BS:IF BS<255 THEN POKE RAM+7,0:
POKE RAM+8,BS
710 IF BS>=256 THEN POKE RAM+7,BS/256:POKE RAM+8,0
720 ? "DRIVE PRESENT?";DP$;:? "";
723 GET #1,X:IF X=89 THEN ? "YES":POKE RAM+9,255:GOTO 730
725 IF X=155 THEN GOTO 730
727 ? "NO":POKE RAM+9,0:DP$="NO"
730 RETURN
733 REM
735 REM *** PRINT RESULTS ***
740 DRIVE=PEEK(769)
750 TR=PEEK(RAM+1):TBL$(1,1)=CHR$(TR)
760 SR=PEEK(RAM+2):TBL$(2,2)=CHR$(SR)
770 IF SR=0 THEN SR=30:GOTO 810
780 IF SR=1 THEN SR=20:GOTO 810
790 IF SR=2 THEN SR=12:GOTO 810
800 IF SR=3 THEN SR=6
810 ST=PEEK(RAM+4):TBL$(4,4)=CHR$(ST):IF ST<256 THEN TBL$(3,3)=CHR$(0)
820 NS=PEEK(RAM+5):TBL$(5,5)=CHR$(NS)
830 IF NS=0 THEN SIDES=1:GOTO 850
840 SIDES=2
850 IF PEEK(RAM+6)=0 THEN DEN$="SINGLE":POKE RAM+6,0:TBL$(6,6)=CHR$(0):GOTO 870
860 DEN$="DOUBLE":POKE RAM+6,4:TBL$(6,6)=CHR$(4)
870 BS=PEEK(RAM+7)*256+PEEK(RAM+8):TBL$(7,7)=CHR$(BS)
880 DP=PEEK(RAM+9):IF DP=0 THEN DP$="NO":GOTO 900
890 DP$="YES"
900 ? :? "DRIVE=";DRIVE
910 ? "TRACKS=";TR
920 ? "STEP RATE=";SR
930 ? "SECTORS/TRACK=";ST
940 ? "NUMBER OF SIDES=";SIDES
950 ? "DENSITY=";DEN$
960 ? "BYTES/SECTOR=";BS
970 ? "DRIVE PRESENT=";DP$
980 RETURN
990 DATA 104,76,89,228
1000 DATA 104,76,116,228
1010 GET #1,CHAR:? CHAR:GOTO 1010
```

APPENDIX C

C.2 GTRPM.COM A Speed Checking Utility For The Indus GT

One of the most common sources of disk drive reading and even formatting problems has to do with the disk drive spinning the diskette at an incorrect rotational speed. This can cause various types of read errors (G5, G4, etc), format errors (F9, etc.) and can even cause software compatibility errors with certain protection schemes whose sector interleave timing requirements are exceptionally stringent. Finding out that your drive is operating at the correct number of revolutions per minute is a good preventive maintenance practice -- like seeing your dentist twice a year.

The GTRPM Utility can be operated either as an extended command from the DOS XL Menu or from the Command Processor level. In either case, all you need to do is type:

GTRPM

and then press <RETURN>. The next screen you will see gives you several choices; you press:

OPTION - To Re-Boot Your Diskette

SELECT - To Tell The RPM Tester You Wish To Test Another Drive

DRIVE NUMBER - To Tell The RPM Tester Which Drive You Wish To Test

The drive number you select must be powered on, connected to the computer, and contain a formatted diskette. Once you have selected a valid drive number and pressed the <RETURN> key, you will be notified that the program is "READING DRIVE NOW". The GTRPM program determines the rotational speed of your diskette (and drive) by reading a single sector's address mark 100 times and then averaging the number of revolutions per minute to which this equates. The tones of the reminding beeps and the colors of the screen will correspond to evaluations of the rotational accuracy reports of the program which will be noted as follows:

288 +/- 1.5 RPM = PERFECT

288 +/- 4.0 RPM = SPEED IS OK

288 +/- 4+ RPM = TOO FAST/TOO SLOW

NOTICE

Indus System Inc. reserves the right to make improvements in the product described herein at any time and without notice.

DISCLAIMER

Indus Systems Inc. shall have no liability or responsibility to the purchaser or any other person or entity with respect to any liability, loss or damage caused or alleged to be caused directly or indirectly by this manual or it's use, including but not limited to any interruption in service, loss of business and anticipatory profits or consequential damages resulting from the use of this product.