MYDOS Version 4 User Guide
Revision 4.3A
for Atari Home Computers

Charles W. Marslett

WORDMARK Systems
2705 Pinewood Dr.
Garland, TX 75042

October 21, 1986

MYDOS Hardware Compatibity

The MYDOS 3 and 4 Disk Operating Systems are intended to work with as many different hardware configurations as possible: either may be used with any Atari 8 bit computer from the oldest 400 and 800 models (with the A Revision OS ROMs) to the current 65XE and 130XE models.

I have verified that the following perpheral hardware works with both major versions of MYDOS:

Atari 810 Disk Drive
Atari 810 Disk Drive with Happy upgrade
Atari 1050 Disk Drive in single density mode
Percom RFD series Disk Drives
SWP Microcomputer Products ATR8000
SWP Microcomputer Products ATR8500
Indus GT Disk Drive (850 and Percom compatible modes)
Z-Tec Hard Disk Interface and Drive

Atari 850 serial/parallel interface
ICD P:R:Connection serial/parallel interface

Axlon RAMPower 128 Plus RAM upgrade for the 800
ICD Rambo XL 256K RAM upgrade for the 800XL
Newell Industries 256K RAM Upgrade for the 800XL
Magna Systems 512K RAM Upgrade for the 800

The following hardware has been reported to work properly with MYDOS:

Atari 1050 disk drives with the ICD USDoubler
Astra Big D (dual double sided disk drive)
Astra 2000 (dual single sided disk drive)
Amdek 3 1/2" systems (with and without 5 1/4" drives)
Indus GT Disk Drive (with new ROM) in all densities
Supra Corp. Winchester disk interfaces and systems
SWP Microcomputer Systems Hard Disk Subsystem

The following hardware or modifications are reported to not work properly with MYDOS 3 and MYDOS 4:

Basic/XE with the Newell 256K Upgrade (prior to 4.2)
Basic/XE with the ICD Rambo 256K Upgrade (prior to 4.3)
Some Indus GT disk drives (perhaps early ones) do not properly recognize diskette densities
Some Percom disk drives (probably the newest ones) do not properly recognize diskette densities
The original public domain 256K RAM upgrade (since it uses a 32K page size) is incompatible with the MYDOS RAMdisk code
Many menu programs and programs with overlays that execute subroutines within DOS 2.0, DOS 2.5 or SpartaDOS.

Summary of changes:
MYDOS Release 4.3A

October 23, 1986

This release of MYDOS, dated October 21, 1986, adds two new commands ('S' to set the RAMdisk drive number and 'V' to turn on and off Write-Verify). It also corrects a long standing problem that occurs if a directory is deleted and includes new code to cycle through all active drives looking for DUP.SYS. MYDOS 4.3 corrected two problems in the error handling code present in earlier versions and had improved manual coverage for several newer features.

The new commands are subsets of the 'O' command permitting the most common changes to be made without reconfiguring the entire system.

The error in the delete code for directories caused the boot sectors and the first file on the disk to be distroyed when the first file was written to the disk after a directory had been deleted (the VTOC free sector bit map was incorrectly updated when the directory was deleted).

The original code to load DUP.SYS ignored errors, now it cycles to the next valid drive on the system if a valid DUP.SYS is not found. Until the CRC/checksum code is added next year, this will cause real confusion if disks with different versions of MYDOS (or Atari DOS) are in use at one time. This permits DUP.SYS to be loaded from a floppy if the DUP.SYS on the hard disk or RAMdisk is inadvertently deleted and you then attempt to go to MYDOS from BASIC or whatever.

The documentation has been extended to cover RAMdisk usage and incorrect information in the error descriptions and configuration command descriptions have been corrected. A short chapter has been added to cover the operation of the RAMdisk driver -- Now the short RAMdisk sequences supported are:

0 -- An Atari 130XE or compatible RAMdisk without BASIC/XE or other XE programs
1 -- An Atari 130XE compatible with 192K for the RAMdisk, the last 64K of which may be used by BASIC/XE, using the Newell Industries memory map
2 -- An Atari 130XE compatible with 192K for the RAMdisk, using the ICD memory map (new) -- in 4.3A this is now BASIC/XE compatible (a 128K RAMdisk and 64K for BASIC/XE)
5 -- An Axlon compatible RAMdisk of at least 40K

The next release (probably in the spring of 1987) will add a CRC check on resident code to guard against errors that might occur after part of the resident code is distroyed. It will also include permanent drivers for the Atari 850 and ICD's P:R:Connection. I may be able to add support for the Atari 1030 and DOS 2.5's extended file system (and I may not!).

Charles Marslett

CONTENTS

MYDOS Version 4 User Guide

by Charles W. Marslett

I. INTRODUCTION

The disk operating system described in this manual is modeled after the ATARI (trademark of ATARI Corp.) disk operating systems (DOS and DOS 2) and may be considered an extension of the very "user friendly" concepts introduced with those two operating systems. The ATARI 810 disk drive and the ATARI 1050 disk drive are well supported by the DOS 2 and DOS 2.5 operating systems, but those DOSs have very limited provision for higher capacity double density disks.

II. SYSTEM REQUIREMENTS

MYDOS 4.3 is intended to function as nearly as possible like ATARI DOS 2. This means it occupies as little memory as possible, supports all the system calls supported by DOS 2 and in most cases, uses exactly the same parameters to the system calls. Most programs that can be copied to a standard ATARI DOS 2 disk and run from that disk can also be run from any MYDOS disk. The largest groups of programs that do not work this way are those that access the disk through calls to entry points inside the DOS file manager or "FMS". Programs that depend on an exact screen menu or a precise format for the directory data will also have difficulty running properly -- one common problem is for programs to expect exactly 3 digits in the file size field and the free sectors field of directories.

The hardware needed to run MYDOS 4.3 consists of at least one ATARI DOS 2 compatible disk drive or equivalent and an ATARI 8-bit computer (400, 800, 1200XL, 600XL, 800XL, 1450XLD, 65XE or 130XE). MYDOS 4 is not compatible with the ATARI 16-bit computers. MYDOS 4 does not support the DOS 2.5 enhanced density format, the DOS/XL Version 4 format or the SpartaDOS format, either.

In addition, if the dynamic density selection or capacities of other than 720 sectors per disk are to be used, the controller must provide the extended 810 interface described in Section 10. This extended interface was first used by the PERCOM dual density disk subsystems, and extended to support 8" and double sided drives on the ATR8000 disk/printer/RS232 controller manufactured by SWP, Inc. Recently support for large RAMdisks and hard disk drives have been added to the group of supported devices by interpreting some of the fields differently. Descriptions of the new interpretation to support most of the hard disk and RAMdisk systems manufactured for the Atari 8-bit computers.

Up to eight disk drives (or nine if the ninth is a RAMdisk) may be accessed, but only one is required. The resident part of the operating system supports all documented functions of the DOS 2 operating

system, so MYDOS 4 supports most available software for the ATARI home computers.

The memory available to a program is affected only by the number of files to be open concurrently: each disk file that may be open at the same time requires 256 bytes of buffer space. Memory requirements are independent of the number of disk drives or the sector size and density. The need for a 256 byte buffer for each concurrently open file means that programs that keep large numbers of files open will have less memory available than under DOS 2 and as a result some programs (mostly older versions of compilers and language interpreters) will not have enough memory to run. With three file buffers and no resident drivers (such as for RS232 support), MYDOS 4 permits binary programs to load as low as $1F00.

MYDOS 4.3 also contains a very versatile RAMdisk driver than can be configured to support most available memory expansion products for the Atari 8-bit computers (the primary requirements are that the memory be accessed through a 16K memory window from $4000 to $7FFF, and that the mapping function be similar to that of either the Axlon or 130XE extended memory).

III. MENU FUNCTIONS

The menu provided by MYDOS 4 identifies 20 common tasks that might need to be done. Rather than having to write a utility program (only a few lines of BASIC would perform most of the menu functions) or even remember the name and format of a DOS command, these tasks can be handled by entering a single letter. MYDOS 4 responds with a question asking for the details of the operation (which file, what density, "are you sure?" or whatever else it might need to know). After you enter the remaining information, the function is performed and another prompt is displayed.

You should notice two interesting things about the menu: the second line on the screen identifies the disk drives present on the system and what they appear to MYDOS 4 to be (single or double density, RAMdisk or high capacity hard disk). The next line describes the current default directory (that directory used when a disk is referenced by 'D:' (without a unit number after the 'D'). The second thing to notice is that after commands fill the screen, the menu 'rolls' off the top: some DOS programs keep the menu, MYDOS 4 does not. This permits more information to be shown on the screen when a long sequence of commands is needed to perform a function or when a directory is being listed or several files are being copied. To restore the menu to the screen, just type RETURN and the initial screen is restored.

Some commands require further information to prevent accidental damage to your disk files: the 'I' command and the 'J' command both require confirmation (through an additional key entry) before distroying the destination disk. To abort either operation without damaging any existing disk files, simply press the RESET or BREAK keys. The 'D', 'E',

'F' and 'G' commands (delete, rename, lock and unlock) all require an explicit file specification. All other commands assume the drive containing the default directory or all the files in the default directory (depending on whether the command affects an entire drive or a set of files).

Disk drive specifications and file specifications are made using the same rules: if only a drive is specified and file data is required, all files on that drive (or in the case of the 'K' 'L' and 'N' commands to save and load programs, the first file on that drive) will be the assumed choice. A drive is specified with a ':' (meaning the default drive), a number (with or without a trailing ':') or the capital letter 'D' followed by an optional number and a required ':'. If you wish to specify the file or set of files to be referenced, the drive format must include a ':' or it must be omitted entirely -- D1:Test.obj, 1:TEST.ASM, or D2TEST (really D1:D2TEST) are valid file names, but d1:Test.obj or 1TEST.ASM are not.

The file name itself is either fully specified (referring to exactly one file on the disk) or includes "wild card" characters (specifying a set of zero or more files). A fully specified file name consists of one to eight characters followed by a period ('.') and zero to three additional characters. The first character in the file name must be an upper or lower case letter, an underscore ('_') or the characters '@' or ''. The '' is the ATARI diamond graphic. The remaining character may be in that set or one of the digits 0-9. The "wild card" characters are the characters '*' and '?': the character '*' or the sequence '*.' end either the 8 character or the 3 character field in the file name and match all possible characters. The character '?' matches any single file name character.

In addition to the main directory (containing up to 64 files or directories) each MYDOS 4 disk may also contain additional directories of 64 files each. If the main directory contained the directory file BAS and the file GRAPHIC1 were in the directory file BAS, it could be referenced with the filename, BAS:GRAPHIC1. If instead, GRAPHIC1 were in the directory GR.dir which in turn were in BAS, then the reference would be to BAS:GR.dir:GRAPHIC1 (and so on with as many names as needed). Because there is no limit to the number of directories on a disk (other than the buffer size of programs using the directories and number of available sectors on the disk), a single diskette can contain hundreds of files if necessary.

Each directory is a 8 sector file to its parent directory. Thus it is limited to 64 files or subdirectories exactly like the 8 sector root directory.

If a disk directory includes the files TEST.ASM, TEST.OBJ, TEST.C, TEST.ALM, TEASET.DOC, TRACE.FIL, and BETS.LST, the specification "t*.*" will not match any file name (since "t" and "T" are not the same letter to MYDOS 4). The specification "T**" will match all but "BETS.LST" (since the others all begin with the letter "T"). The specification "?E??.*" will match the first four files and the last one (since the 8 character part of the file name must have no more than 4 characters in it and the second character must be an "E"). The specification "*?" will

match only the file TEST.C (since it is the only file name with a single character in the 3 character field). The specification "????E**" will match the files TEASET.DOC and TRACE.FIL and none of the others (since the 8 character part of the file name must have at least 5 characters and the fifth must be an "E").

Where more than one file name is asked for, the first may be omitted by starting the response with a space or comma, and the last may be omitted by ending the line with a comma (the space cannot be used here since trailing spaces are ignored). If both file names are entered, they may be separated with either a space or a comma. Some commands may be modified using a letter following the character '/' after the file name (for example, 1/A or D1:TEST/A). The letter used (the modifier) generally means the same thing if it is allowed. Invalid modifiers are always ignored with no error indication. The modifier '/A' causes the results to be appended to the end of an existing file. This is applicable to 'C' (copy) and 'A' (directory) commands. The modifier '/N' causes the destination disk formatting to be skipped (saving about a minute) when used in the 'I' (initialize) and 'J' (duplicate disk) commands. It prevents the questions asked before changing each file if it is used in the 'D' (delete), 'F' (lock) and 'G' (unlock) commands. (We bend the rule so recently mentioned. In both cases, part of the function is skipped see?)

The '/X' command causes MYDOS 4 to pause at the end of each read or write pass when copying data to allow you to change disks (permitting you to copy from one disk to another with a single drive, even if that drive appears to MYDOS 4 to be two drives). This option supports disk drives that handle increased capacity by making a single diskette appear to be on more than one drive at a time, as well as disk drives that select density through the drive number (1-4 are single density, 5-8 are double, for example, and drives 1 and 5 are the same physically). This also allows the writing of a directory of one disk onto another as a file (use the command 'A' followed by the entry "1,1:DRV01.dir/X") even with only a single drive on the system. The '/X' is assumed if only one file name is entered in the copy command. This emulates the operation of the ATARI DOS 2 'O' command which is not implemented in MYDOS 4.

To omit copying files with extensions beginning with 'S', the '/S' modifier can be appended to the either file specification in the 'C' (copy) command: for example, the line "2/S,1" will copy all files not matching the string *.S?? from drive 2 to drive 1.

IV. THE MENU COMMANDS

A. List a Directory or a Set of Files

The 'A' command will list the files on a disk with their sizes, followed by a line specifying the number of free sectors on the disk. If the line starts with an '*', the file has been locked and may not be modified or deleted without first being unlocked. A ':' before the file name marks those files that are directories. These files cannot be read or written as other files but only accessed as directories or deleted (if the directory

is empty). File lengths and the number of free sectors are reported as 4 digit decimal numbers (most other Atari DOSs report the values as 3 digit numbers).

If the number of free sectors on a drive exceeds 9999, the free sector line will contain a 5 digit number, and its length then will be the same as the file data lines -- this may cause problems with some programs that do not detect this line by examining the first character (a file line if it is '0' or '*' and the free sector line if it is a digit). Also, if the number of sectors in a file is greater than 9999, there will be no blank between the 3 character file extension and the file size (so the line will remain the same length). This was needed to retain the double column directory list, AND DOES NOT APPLY IF A DESTINATION FILE NAME IS SPECIFIED (or if the directory is read from BASIC or from a program). Directories read from programs will get four or five digit file sizes and some directory lines will be one character longer if any such huge files exist on the disk.

No indication is made of the format of the file but ATARI DOS 1, ATARI DOS 2, and MYDOS 4 are the only three supported file formats, and the DOS 1 format will be dropped with the release of MYDOS 4.4. SpartaDOS, DOS/A+ Version 4 or Atari DOS 2.5 files must be converted to single or double density Atari DOS 2 files first in order to be usable from MYDOS 4.

See Section 6 for further directory information if you need more detail than the 'A' command provides.

This command will list the directory information to the screen if only one file specification is entered. If two are entered, the second is taken as a destination file and will be overwritten (or appended to if the /A flag is used) with the directory data. The entry "1,P:" will write the directory of the disk on drive 1 to the printer, for example.

To list the files in a subdirectory, enter the name of the directory followed by a colon -- ':'. For example, "1:TEST:BAS:" will list the files in the directory BAS which in turn is in the directory TEST in the main directory of the diskette in drive 1.

B. Run the Cartridge

The 'B' command returns control to the cartridge in the left (or only) cartridge slot. If no cartridge is present, an error is displayed, and nothing happens. No additional information is required, so if a cartridge is present it is entered after loading MEM.SAV (if the last load command were an 'N') or immediately (if the last load command were an 'L').

C. Copy a File or a Set of Files

The 'C' command is used to make another copy of one or more files of data. The two file specifications asked for after entering the 'C' identify the source and the destination of the information being copied. Either may be fully specified disk file or a device specification (such as E:, P: or one of the RS232 ports R1: to R4:). The destination may be a set of disk files (specified with '*' and '?'s) only if the source specifies a file name for the destination to use.

Copies from a file set to a device will implicitly write consecutive files to the device (generating a set of listings or a collection of cassette files for example). The source may be a set and the destination a single disk file, but unless the '/A' modifier is specified to append each copied file to the end of the previously copied files, only the last source file will remain on the destination disk.

Note that the 'C' command always uses the full memory space for a copy operation (unlike ATARI DOS 2) and as a result, it will always invalidate MEM.SAV if it is used. Any pending program cannot be restarted after a 'C' or 'J' command.

D. Delete a File or Set of Files

The 'D' command will remove all files that match the file specification entered asking for confirmation before each one is removed. This verification that the file is really the one to be removed can be disabled for the duration of this single 'D' command by adding the command modifier '/N' to the end of the file specification. In this case, all the matching files will be removed 'quietly' and the only further indication you will see is the prompt for the next command.

E. Rename a File or Set of Files

The 'E' command changes the name of the source file or files to match the specification in the destination. Unlike other file specifications, the destination specification must consist of a single file name: it must not contain any directory names or a disk drive number. For example, "D2:TEST:BASIC:NOTPNT.BAS,RANDIO.BAS" is the line entered to change the name of a file in the directory "D2:TEST:BASIC". To change the name of the directory "BASIC" to "ATBASIC", the line would look like "D2:TEST:BASIC,ATBASIC".

F. Lock a File or Set of Files

The 'F' command limits access to the files identified. The files may not be deleted, renamed, added to or replaced without being first unlocked with the 'G' command.

When a directory is listed, the files that have been locked using either the 'F' command or the 'lock' or 'open locked' functions provided through CIO will be marked with an '*' in the first column. The files that are locked may be read or loaded and executed normally, only modification or removal are prohibited.

A locked directory cannot be deleted or renamed and no files may be written to it. The files already in it can be modified freely, however.

Before each file is locked MYDOS asks you for confirmation with a message: for the file TEST, the message would read "Lock TEST?". Any answer but 'Y' will result in the file not being locked. The confirmation questions can be skipped by adding '/N' to the end of the file specification.

G. Unlock a File or Set of Files

The 'G' command removes the limitations imposed on a file when it is 'locked' using the 'F' command. It does not alter the file or otherwise change the way the file is accessed or used. The same function may be performed in a program through the CIO function to 'unlock' a file.

Before unlocking each file MYDOS asks for confirmation with a question that must be answered with a 'Y' if the file is to be unlocked; otherwise, no action is taken and the next confirmation question is asked.

To disable the confirmation questions, enter '/N' after the file specification (see Section IV.D., on deleting files, for a more detailed explanation).

H. Write MYDOS 4 to a Disk

The 'H' command is used to make a rebootable copy of the current MYDOS 4 files in memory. The two files created or rewritten are 'DOS.SYS' and 'DUP.SYS'. 'DOS.SYS' is an image of the permanently resident file management routine accessed through CIO and the small interface package that loads and saves MEM.SAV (an image of the part of memory used to hold the nonresident part of MYDOS 4) and the second part of MYDOS 4 itself ('DUP.SYS'). The file 'DUP.SYS' is a standard load file containing the part of MYDOS 4 that is overwritten when a program is loaded into memory. Neither of these files is compatible with any other disk operating system either for the ATARI or any other home computer. Both should be treated as a single object. Never copy only DOS.SYS or only DUP.SYS to a disk without copying the other.

The files written to the disk by the 'H' command will reflect the configuration parameters currently in memory, which may be different from the ones active if the system were rebooted from the master disk

again. (See Section IV.O., configuring the system, for the definition of the configuration parameters provided in the system and how to specify a modified configuration.)

I. Initialize a Diskette

The 'I' command is used to prepare a new disk for use with the MYDOS 4 operating system or to remove all the files on an old disk. The result of the 'I' command is a completely empty disk. The only data on the diskette is that system provided information defining the space available and the empty main directory.

If the drive number is followed by a '/N' modifier, the diskette will not be reformatted, but just 'erased'. If the disk is formatted with a standard format, this is the recommended (fastest) way to remove all the files on a diskette, rather than to use the 'D' command.

A diskette may also be formatted in the Atari 1050 enhanced format by responding to the question "Enter Y (or A) to format drive 1:" with an "A" (presuming the drive is a 1050 drive or compatible with the 1050). The diskette is, however, not compatible with the diskettes similarly created by DOS 2.5. The MYDOS diskette will not be readable by DOS 2.5 and DOS 2.5 enhanced disks can be only partially read by MYDOS.

J. Duplicate a Diskette

The 'J' command copies all the information from one diskette to another. The information to be copied form the source diskette is determined by specifying a starting and an ending sector number. If the range is not stated, the sectors marked as in use in the DOS bit map (on the VTOC sectors) are copied.

A sector range is specified by adding two numbers separated by a dash and enclosed in parentheses to the end of the drive specification(s). For example, to copy sectors 19 through 54 (tracks 1 through 3) from drive 1 to drive 4, without formatting the diskette in drive 4, the command line could be "1,4/N(19-54)".

If the destination disk is already a properly formatted MYDOS 4 diskette, the '/N' modifier may be entered after either drive number to skip the formatting of the destination drive. Otherwise, the destination diskette will be formatted before the data from the source is copied to it. That is, either "1/N,2" or "1,2/N" will copy from drive 1 to drive 2 without first formatting the diskette in drive 2. To copy the first two tracks of a diskette without formatting the diskette being copied to, you could enter "1/N,2(1-36)" or "1,2/N(1-36)".

If the VTOC is not constructed by Atari DOS 2, a version of MYDOS or another compatible operating system the sector list must be specified. A dual density Atari DOS 2.5 disk will not be copied correctly unless the

destination is first formatted with the 'I' command, then copied with the 'J' command and the command line "1,2/N(1-1040)".

To copy the entire disk from drive 1 to drive 2 after formatting the diskette in drive 2 the command could be "1,2(1-720)". This is the proper way to duplicate a non-MYDOS compatible diskette (of course, copies of "copy protected" disks will not normally be usable after copying but data disks for some word processors and games can be backed up this way).

The disk initialization done by the 'J' command is done without error checks: this means that a disk formatted with the 'J' command may have bad sectors (in the case of creating a backup disk, the disk will not be written to later so if the disk is written with write verification any error in the current contents will be discovered before modifying the source disk).

If the disk is to be a working disk, a more reliable approach is to initialize the disk (with the 'I' command) and then copy the data using the 'J' command and the '/N' modifier if no bad sectors are identified (see the next paragraph). Since the 'C' command reorganizes the data on the disk, reducing sequential read time for all the files, it is a better way to copy the contents of disks when the read time on a diskette is more important than the copying time. If a diskette with DOS.SYS and DUP.SYS is copied with the 'J' command, the copy will also be bootable but the 'H' command must be used to reinstall a bootable DOS.SYS if the diskette is built with the 'C' command.

Note that the 'J' command, like that in ATARI DOS 2, will use all of available memory to duplicate the diskette: this means that if memory has been saved using the MEM.SAV file, it will no longer be valid. Any pending program cannot be restarted after a 'C' or 'J' command.

K. Save Memory to Disk

The 'K' command builds a binary load file containing the data from the memory area specified, as well as an initialization and a run vector address if specified. If the file is not to execute an initialization routine on being loaded, the initialization vector should be omitted. If it is not to run on being loaded, the run vector should also be omitted (trailing commas need not be typed in either).

If either vector is entered as zero, that is equivalent to omitting it. Note that the starting and ending addresses of the program and both entry points are all specified as hex numbers.

If MEM.SAV is active when the 'K' command is entered, the MEM.SAV file is loaded before writing the file to the disk.

L. Load Memory from a File

The 'L' command takes a binary load file from the disk and loads it into memory. The load file's initialization routine(s) will be executed and the program started at its run address unless the '/N' modifier is appended to the file name given the 'L' command.

This command disables the MEM.SAV file before loading and executing the program.

M. Run at Address

The 'M' command is used to enter a program loaded without a run address, or to jump into any program without the need for a return address. It may be used to restart the computer (loading the AUTORUN.SYS file, if any) by specifying $E477 as the jump address.

If MEM.SAV is active (enabled with the 'N' command and not since disabled by the 'L' command), the contents of memory will be restored from MEM.SAV before jumping to the address specified.

N. Load MEM.SAV from a File

The 'N' command takes a binary load file from the disk and loads it into memory. The load file's initialization routine(s) will be executed and the program started at its run address unless the '/N' modifier is appended to the name of the file to be loaded. This command enables the MEM.SAV file before loading (and executing) the program and when control is returned to MYDOS the contents of memory will be saved back to MEM.SAV.

If no file name is specified, the MEM.SAV file usage is enabled but no program is loaded or run.

O. System and Drive Configuration

The 'O' command is used to specify the type (at least logically) of the disk drives on the ATARI computer and it is also used to specify several system configuration parameters such as the number of file buffers provided or whether a RAMdisk is present. The system configuration functions, which are not specific to individual drives, are selected by entering a RETURN when the prompt asking for a drive number is displayed.

Either three or seven configuration parameters will be prompted for during a system configuration: three if the RAMdisk (also referred to as memory or virtual disks) is not to be used or seven if the MYDOS 4 RAMdisk handler is to be used.

If the answer to the third question is 'N', the remaining four questions having to do with the RAMdisk will be skipped. The following table explains the significance of each prompt and its response.

Verify WRITEs?	N=do not write with verify, else do verify at once (RETURN = 3)
RAM disk present?	N=no RAM disk on the system (skips the next 4 questions as well), else the RAMdisk is enabled and the next 4 responses determine what kind it is
Size(K)?	Size of the RAMdisk area in K (8 single density sectors = 1K), this must be a multiple of 16 (NI 256K upgrade=192 OR 128, AXLON=112, default or 130XE=64).
Control Address(HEX)?	Address of the memory map control port: a RETURN specifies $D301 for the 130XE. For AXLON RAMPOWER 128, enter CFFF here.
Page sequence?	RETURN if using a standard Atari 130XE or equivalent RAMdisk. Other standard values are (1) for the Newell industries upgrade, (2) for the RAMBO from ICD (only if no extended RAM programs ar being used: BASIC XE, SynCalc, etc), and (5) for Axlon or compatible boards. The most general response is to list the page numbers to be used as 2 hex digits each, separated by commas, and continuing if necessary by ending the line with a comma. The sequence is ended with a "0" if the RAMdisk is 130XE compatible and with a "FF" if it is Axlon compatible.
RAM disk drive no?	Drive number used to access the RAMdisk (RETURN for drive 9 or a drive number 1-9)

A RETURN does not retain the current value when responding to these questions, it changes the setting to a generally acceptable default.

Also, a BREAK or RESET will not leave the configuration unchanged. If the configuration process is aborted for any reason you should reboot the computer or properly complete the configuration before doing anything else with the computer.

If instead of a RETURN, a drive number or name had been specified then that drive would be reconfigured. The first question identifies whether the drive is to be included in system initialization (and thus be available for later use). If a non-existent disk is included it does not cause any problems with the system: it simply causes that disk to be examined each time the system is booted (adding perhaps a second to the time it takes to boot MYDOS 4).

If the drive is excluded from the system, no further questions are asked. Otherwise, the second question asks if the drive is configurable: is it like the ATARI 810 drive (with a fixed configuration) or is it like the PERCOM or ATR8000 drives. If the disk is not configurable it is assumed to be a 720 sector, single or double density ATARI 810/815/1050 disk drive.

The third question is whether the drive is a high capacity drive (does it use the modified configuration for non-floppy disks). Generally this should be answered 'N' if the drive is a floppy disk drive and 'Y' otherwise. If this question is answered 'Y' the only remaining question is the size (in sectors, from 256 to 65535, of the drive).

Drives up to 16 million bytes (Megabytes) capacity can used with MYDOS, and larger drives can usually be partitioned to appear as two or more drives of less than 16 million bytes capacity.

Drives excluded from the system can be dynamically added by referencing them but they will always be treated as 5 1/4 inch 810 compatible drives (the default configuration).

If the first three answers are 'N' (do not exclude the drive), 'Y' (it is configurable), and 'N' (it is not a "smart" or high capacity drive), the configuration is asked for: Is the drive double sided, how many tracks are there on the each side of the disk, and at what speed can it move the read/write head across the disk (what is its step rate).

The first question is answered with 'Y' or 'N' ('Y' meaning 'yes' it is a double sided drive and diskettes formatted on it will be double sided).

The second question is answered with 35, 40, or 80 followed by a RETURN if the disk drive is a 35 track, 40 track, or 80 track 5 1/4 inch floppy drive and with 77 if the drive is a 77 track 8 inch drive. If you have IBM PC/AT style high capacity disk drives (1.2Mb capacity) you may use 77 of the 80 tracks in high capacity mode providing 1 Mb capacity by selecting 77 tracks. No other numbers are accepted. The answer to this question specifies both the type of drive (8 inch/high capacity 5 1/4 inch, or standard 5 1/4 inch) as well as the number of tracks per inch and total capacity of the drive. This answer is very important to the operation of the drive.

The last answer is entered as a code: use the following table and the drive specifications to determine the proper value.

Code value	8 inch rate	5 1/4 inch rate
0	3 ms/track	6 ms/track
1	6 ms/track	12 ms/track
2	10 ms/track	20 ms/track
3	15 ms/track	30 ms/track

P. Diskette Density Selection

The density used for most MYDOS commands is determined by the data written on the diskette and the operator need not worry about setting it. The 'P' command is provided to allow forcing the density setting for the format ('I') command and to permit setting the density for disk drives and systems that do not automatically recognize the density of a diskette when sector 1 is read (for example, some Percom and Indus GT drives).

MYDOS commands that access a diskette will automatically select the appropriate density, so the 'P' command will have no effect on the drive if any command accessing the drive configured with the 'P' command is executed before the format ('I') command or if a program is run using the 'B', 'L' or 'N' commands.

MYDOS has an untested feature to support the Atari 1050 enhanced density mode -- when formatting a diskette, answering "A" rather than "Y" to the last prompt will cause MYDOS to issue an enhanced density format command and will cause the disk sector map data to be written so as to allow use of all 1040 sectors on the disk.

(Under normal operation, MYDOS does not need to know if the diskette it is reading or writing is formatted as enhanced or standard density. The diskette is just a 1040 sector single density diskette.)

Q. Create Additional Directories

When a diskette is formatted, an empty directory (the highest level or root directory) is created. This directory is capable of holding up to 64 files or other directories. If additional directories are installed in this directory, each of the additional directories can contain up to 64 files as well. A directory is installed in an existing directory using the 'Q' command and responding to the question of what the directory name is with the name of the new directory.

For example, if "TEST" and "BAS" are two directories in the root directory of the diskette in drive 1, "1:TEST:COMM" or "1:BAS:COMM" would create a new directory in "TEST" or "BAS" respectively. "1:NEW:COMM" would not create a directory, however, if the directory "NEW" does not already exist.

A 'Q' command with the response "1:NEW" would create it, and a second 'Q' command with the response "1:NEW:COMM" would then create the nested directory.

Each directory takes up exactly 8 sectors and after it is created it may only be referenced as a directory (followed by a ':' that is) or deleted. And it may only be deleted if it is empty (if it has no files in it). A directory may be emptied by using the 'D' (delete) command and specifying the files "*.*" or "*.*/N" to remove all the files in the directory.

You can most safely do this by specifying the full name of the directory: for example,

D5:MSGBASE:SEPT1983:*.*/N

R. Set the Default Directory

The 'R' command is used to select a directory to be used when a file is referenced without the drive number: that is, when file names such as "TEST1.BAS" or "D:NEWCODE" or even ":BIGFILE" are used, they are assumed to be in the default directory. Programs run under MYDOS 4 can access the contents of the current default directory by using a file name of the form "D:..." without a drive number explicitly entered.

The directory is set by inserting the diskette with that directory on it into the desired drive, then entering the file name of the directory with no trailing ':'.

The program may also set the default directory by calling the CIO Function code 41 (set directory) routine.

If the diskette in the drive containing the default directory is replaced, or if the default directory on a diskette is deleted, the default should be redefined. The one exception is if the default directory is the root directory on a diskette: because only the root directory is at the same location on all diskettes.

S. Set the RAMdisk Drive Number

The 'S' command permits the drive number assigned to the RAMdisk to be changed easily, without reentering the full system configuration, using the 'O' command. In response to the prompt, just enter the new drive number and press the RETURN key.

V. Set Write Verify ON or OFF

The 'V' command turns on or off verification (read back) of data written to disk drives (other than the RAMdisk). MYDOS does not support a true file data verification. The hardware verify function does, however, insure that the data written can be read back an instant later.

This is much more reliable than no verification at all and it will catch perhaps 80% of all failures to write the sector data properly to the disk (drive speed problems and incorrectly seated diskettes will not be caught).

V. FILE MANAGER FUNCTIONS PROVIDED THROUGH CIO

MYDOS 4 supports all CIO calls supported by ATARI DOS 2, with modifications to the OPEN (Function code 3) and the FORMAT (Function code 254) functions.

Three additional CIO functions have been added: MAKE DIRECTORY (Function code 34), SET DIRECTORY (Function code 41) and LOAD MEMORY (Function code 39).

The OPEN function in ATARI DOS 2 does not use the data provided in the AUX2 byte, but in MYDOS 4, when the AUX1 byte is 8 (the file is opened for creation or replacement), the AUX2 byte contains three flags that control the file format and whether it will be created locked or not.

If AUX2 bit 1 is set, the file will be written in the original ATARI DOS 1 format if the disk is single sided, single density; otherwise, it will be ignored: the original format is not viable for 256 byte sectors or disks with more than 1023 sectors.

If AUX2 bit 2 is set, the file will be written in MYDOS 4 format, and may contain sectors beyond absolute sector 1023. Such a file may not (easily) be read by programs not running with MYDOS. This is the only format used with high capacity disks.

If AUX2 bit 5 is set, the file will be written with the 'LOCKED' bit in the directory set initially. This is provided for use by multi-tasking functions (such as a print spooler, sequential file pre-reading function or other enhancements one might want to make to the standard OS or DOS provided functions).

The FORMAT function in ATARI DOS 2 does not provide for any variations to the standard disk usage: in MYDOS 4, the contents of the AUX1 and AUX2 bytes are used to specify the number of sectors on the disk being formatted and whether the disk needs to be formatted by the controller as well as needing directory initialization. Bit 7 of AUX1 is set to skip the physical formatting of the entire disk surface when it is not required, and bits 6-0 of AUX1 and all of AUX2 can be used to specify the number of sectors on the disk being created (if all 15 bits are zero, the disk is assumed to be the size defined by the drive configuration).

This permits formatting a single sided diskette on a double sided drive, for example.

To load (and possibly execute) a program file, MYDOS provides the CIO function 39 call. From BASIC you can load and execute a program by executing the line:

```
XIO 39, #3, 4, 0, "D:MYPROG.OBJ"
```

Any inactive IOCB can be used, and if AUX1=4 both the INIT and the RUN entries will be executed. If AUX1=5, the RUN entry will be executed, if AUX1=6, the INIT entry will be executed, and if AUX1=7, the file will be loaded without executing either entry point.

Any other values of AUX1 will return an error code and do nothing.

Another XIO call, XIO 34, has been added to create a directory. When a directory is created, the name used must not match any existing file or directory in its parent (for example if the directory to be created is named "D1:TEST:", there can be no other directory in the main directory named "TEST" nor a file named "TEST" there.

From BASIC the XIO 34 call is "XIO 34, #iocb, 8, 0, dirname" where "iocb" is any available unit number, and "dirname" is the name of the new directory (it does not end with a trailing ':').

The final function added to those provided by ATARI DOS 2 is XIO 41, to define the default directory. The default directory is that which will be searched for a file if the file name begins with "D:". In ATARI DOS 2 this default directory is always "D1:" but in MYDOS 4, the default directory can be any root or subordinate directory on any disk in the system.

The buffer address passed CIO in the XIO 41 call is the address of a string that contains the default directory name, terminated with either an end of line ($9B) or a null byte ($00).

The directory will be accessed before returning to the calling program so that an error in specifying the directory will be reported as early as possible.

VI. CIO FUNCTION CODES PROVIDED BY MYDOS 4

Function code 3, OPEN

The open function uses the buffer address to point to an ATASCI string terminated with a character not 0-9, A-Z, a-z, :, ?, or *. This string is the name of the file to be accessed or created. A good terminator for this string is either a null ($00) or an end of line ($9B).

The AUX1 byte defines the usage of the file: 4 for input, 6 for directory data reading, 8 for creating/replacing output, 9 for creating/appending output and 12 for input/update (without extension).

The AUX2 byte is used when a file is replaced or created, and contains three significant bits, bit 1 set causes a DOS I format file to be created if the diskette is single sided, single density (otherwise, it is ignored). Bit 2 set causes the MYDOS format to be used even if the diskette is a 40 track single sided diskette. And bit 6 set results in the file being LOCKed initially without and additional CIO call.

For input, update or directory access AUX2 is ignored, and the length is always ignored. In normal use, AUX2 is set to zero emulating ATARI DOS 2 usage.

MYDOS 4 also does not leave partially full sectors when appending to a file. This has two effects on programs which open files in append mode:

the open will fail if the file cannot be appended to rather than the close (as in ATARI DOS), and the file size will not change if a file appended to is copied to another disk (in ATARI DOS it may grow smaller).

MYDOS 4.3, unlike earlier versions, will be forced to reread the entire file when closing it if more than one file at a time is opened in append mode.

Function code 5, GET RECORD

The get record function reads a line of data into a buffer, the buffer being defined by its starting address and length. The line is defined as the data bytes in the file up to an end of line character ($9B) or until the buffer is full, whichever occurs first. The line is also terminated if the end of the file is read. All record I/O is buffered in MYDOS so record transfers are necessarily slower than unbuffered I/O.

No other fields of the IOCB are referenced or needed. Note that the ATARI ROM OS supports single byte I/O through the accumulator if the buffer length is set to 0. In this case, GET RECORD and GET CHARACTERS function exactly the same way.

Function code 7, GET CHARACTERS

The get characters function reads a fixed number of bytes from a file into a buffer, the buffer being defined by its address and length (two 16-bit number in the IOCB). The only case where the buffer is not always filled is if the end of the file is read.

As is the case with get record calls, a single byte may be read into the accumulator by setting the length field to zero. A get character CIO call will be perform unbuffered I/O if the buffer is longer than 256 bytes (ATARI DOS 2 sets a similar threshold at 128 bytes). For this reason a single long input is considerably faster than several short ones.

Only the buffer address and length in the IOCB are used by the get characters function.

Function code 9, PUT RECORD

The put record command will write a single line to an output file, the line defined the starting address of the buffer and either the length of the buffer if no end of line ($9B) bytes are encountered, or the first end of line byte. Only the buffer address and length in the IOCB are used in this command.

Function code 11, PUT CHARACTERS

The put characters command will write the contents of a buffer defined by its address and length (in the IOCB), to a file opened for output. The entire buffer is always written to the file unless the write is to an output/update file and the end of the file is reached or the write is to an output/append or create file and the last sector on the disk has already been allocated. Only the buffer address and length fields in the IOCB are used when the put character function is called.

Function code 12, CLOSE A FILE

To terminate use of a file (and for an output file, to write the incomplete buffer to the disk) the IOCB used to access the file should be closed. This is done by setting the function code in the IOCB to 12 and calling CIO. The close function does not use any of the data in the IOCB for any purpose whatsoever.

Function code 13, READ STATUS

The read status command is issued to an unopened IOCB, with the buffer address that of a file name string. If the file is not present that error condition is returned, if it is locked, that error condition is returned; otherwise, a normal completion code is returned. Only the function code and the buffer address in the IOCB are needed.

Function code 32, RENAME A FILE

The rename function is passed a character string (pointed to by the buffer address in the IOCB), and the first part of the string is a file name string identifying the file or files to be renamed. Following a single invalid character (one invalid in the file name, that is) a simple file name must also be present: this second file name cannot include any drive or directory names.

An example, using a comma as the invalid character, is

D2:TEST:PGMS:A.OUT,ZCPY

which will change the string needed to access the file "D2:TEST:PGMS:A.OUT" to "D2:TEST:PGMS:ZCPY" -- Note that only the last file name (if subdirectories are used) can be changed, to change "PGMS" to "MLPROGS", the buffer must contain "D2:TEST:PGMS,MLPROGS" and the rename will also change the full names of all files in "D2:TEST:PGMS" (to belabor the obvious).

Wild card characters should appear only in the part of the file name following the last ':' and their effect is best described by an example.

The string "D2:TEST:*.*,*.XYZ" will rename all the files in the TEST directory, making each file's extension ".XYZ".

If the directory had the files "ATEST.BAS", "LOG", and "REPORT.XYZ" in it, the result would be a directory with "ATEST.XYZ", "LOG.XYZ" and "REPORT.XYZ" in it.

Function code 33, DELETE A FILE

The delete function removes any files that match the file name string pointed to by the buffer address in the IOCB. Files locked will not be deleted, so must be unlocked before being removed, and directories that are not empty (that have a file, even an empty file, in them) cannot be deleted. If either case is attempted, the corresponding error code is returned. Otherwise, the files are removed and their data areas are returned to the free space on the disk.

Like other Atari DOSs, in MYDOS files removed cannot be (easily) recovered after being deleted. This is unlike some other operating systems that preserve deleted files for as long as practical before overwriting them.

Function code 34, MAKE DIRECTORY

The make directory function will create a new subdirectory on a disk (it is not used to create the first directory, that is the "root directory" identified by the drive specification "D1:", for example). It is called through CIO by storing the address of the new directory's name in the IOCB buffer address and setting up AUX1 and AUX2 as for an open call (see Function code 3), normally AUX1=8 and AUX2=0.

This function has no effect on the current default directory, and if it is desired to make the newly created directory the default one, the program must make a set directory call (Function code 41) following the make directory call (the order is very important, because the default directory cannot be set to a nonexistent directory).

Function code 35, LOCK FILE

A file can be "locked" so that it may not be modified or deleted inadvertently by calling CIO with the lock function. The buffer address is used to point to a file name string that identifies the files on the disk to be locked. The only file modification that can be performed on a locked file is to unlock it.

The lock function can be requested for a file already locked, and it will return no error (unlike other file modification calls to CIO), but the status of the file will not have been changed either.

Function code 36, UNLOCK FILE

The unlock function is identical to the lock function except that is reenables the modification or deletion of an unlocked file. A file that is not locked can be unlocked with no error returned and no change in the file's status.

Function code 37, POINT TO POSITION IN FILE

The point function is passed the 3-byte disk address to be positioned to in the twelfth through fourteenth bytes of the IOCB. On return, the next byte read from that IOCB will be the one that was read or written next after the corresponding note function was executed. A point call to CIO can only be made if the file can be used for input: that is, if it is opened for input or update processing. The first two bytes of the disk address are a sector number (in low byte/high byte format) and the third is the byte within the sector.

If a file is being appended to (opened with AUX1=9), a point function call made before closing the file may return an unexpected error (this cannot happen with the note function, however).

A problem can occur if the file being pointed to is in the last half of a 16 Megabyte disk: Atari BASICs do not allow sector number to be greater than 32767. A solution is to use the following six line substitute for the POINT statement (with attention paid to the fact the the two AUX bytes must match the two used to open the file):

```
OPEN #K,AUX1,AUX2,"D5:BIGFILE"
NOTE #K,SECTOR,POSITION
. . .

HIGHBYTE = INT(SECTOR/256)
LOWBYTE = SECTOR - HIGHBYTE*256
POKE 844+16*K,LOWBYTE
POKE 845+16*K,HIGHBYTE
POKE 846+16*K,POSITION
XIO 37,#K,AUX1,AUX2,""
```

Function code 38, NOTE POSITION IN FILE

The note function returns in the twelfth through fourteenth bytes of the IOCB a 3-byte disk address that may be used at a later time to reposition the file using the point function. The note function can be used on files open for input, output, update or appending.

The three bytes returned are the low byte of the sector address, the high byte of the sector address, and the byte within the sector in that order.

Function code 39, LOAD MEMORY

The load memory function takes a file formatted in the ATARI DOS 2 executable program format (generated by the "K" command, by the assembler/editor cartridge, by AMAC or MAC65, or by any of several compilers for the ATARI computers) and loads its contents into the computer's memory as specified in the file.

No offset control is provided and no part of memory is protected from the loading process. The initialization and execution addresses (if any) can be individually enabled and disabled, however, to permit loading and patching a program then writing it back to the disk for normal use.

To load a program into memory, the address of the file name string is stored into the buffer address and a value of 4, 5, 6 or 7 is stored into the AUX1 field.

If AUX1 is 4, both the initialization routines and the run address are executed after closing the IOCB used but before returning to the calling program. If AUX1 is 5, the initialization routines are disabled, but the program will be run. If AUX1 is 6, the initialization routines will be run, but the program execute address will be loaded and ignored. If AUX1 is 7, the text of the program will be loaded into memory, but no other activity will be performed.

Function code 41, SET DEFAULT DIRECTORY

The set directory command will use the contents of the buffer as a file name and open the specified file, determining if that file is a valid directory. If so, it will become the new default directory. That is, file names of the form "D:..." will be assumed to be in the default directory (which may be on any disk in the system and may be either the root directory of that disk or a subdirectory).

Only the buffer address and the function code are significant when setting the default directory.

Function code 254, FORMAT A DISKETTE

The format function uses the contents of the buffer pointed to by the buffer address to identify the drive containing the diskette to be formatted. If both AUX1 and AUX2 are zero, the disk is formatted according to the capacity data in the system control table defined using the 'O' command. If AUX2 bit 7 is set to 1, the format operation is skipped and an empty file system is written to the diskette. (This assumes the disk is preformatted.)

The remaining 15 bits of AUX1 and AUX2 are used as a 15 bit number to specify the number of sectors available on the disk (permitting the use of the last few sectors of a disk outside the file system if desired).

VII. DISK STRUCTURES SUPPORTING MYDOS 4

MYDOS 4 uses the first three sectors of a disk to hold some disk information and the initial boot program if the drive contains DOS.SYS and DUP.SYS.

Sector $168 (and sectors $167, $166, $165, etc., if the disk is formatted as a higher capacity disk not compatible with ATARI DOS 2) is used to hold a bit map of available sectors and several flag bytes identifying the default format of files on the disk.

Sectors $169 through $170 contain main disk directory data, identifying the files on the disk, their sizes and their starting sector number.

Note that this usage, when the diskette is a 719 sector volume declared to be DOS 2 compatible, is in fact exactly the same as ATARI DOS 2 would make of the disk. The default single sided format differs only in that sector 720 is not left out of the file system in MYDOS but is used to provide 708 free sectors in an empty diskette rather than 707.

The changes made when the high capacity format is chosen are to allocate enough sectors before sector $168 to assign a bit for each sector that may be allocated for a file or for use by the system. The high capacity disk directory may be read by ATARI DOS 2, but the data in the files can only be accessed if it falls in the first 1023 sectors of the disk and then only if the file number checking code in DOS 2 is disabled.

This format and MYDOS 4 support accessing disks of up to 65,535 sectors of 256 bytes each (approximately 16 Mbytes).

Compatibility with Atari DOSs is further reduced if subdirectories are used: to ATARI DOS 2.0, the subdirectories will appear to be simple files with unreadable contents. The subdirectory's files will not be accessible and the subdirectory can be damaged if it is written to (even by appending). For this reason disks sold to the general public, exchanged with friends, and so forth, should not contain subdirectories unless there is reason to require that the disk be used with MYDOS.

A further problem with exchanging diskettes is that there are many different formats are used by vendors of double sided disk systems for the ATARI. For this reason, double sided disks not only require both computers use MYDOS, but also require that they use the same disk system (PERCOM, SWP, Astra, Supra or whatever).

VIII. MYDOS 4 MEMORY MAP

The MYDOS 4.3 disk operating system occupies the area from $0700 to $1EEE at all times, and when the menu is active, it also occupies the area from $1EEE to $40FF. In addition, the first 16 bytes of the floating point workspace ($D4 - $E3) are used by MYDOS 4 at that time. Unlike

ATARI DOS 2, MYDOS 4 utility program (DUP.SYS) also calls the floating point ROM entry points.

The nonresident part of MYDOS 4.3 starts loading at $26EE, reserving the area from $1EEE to $26EE for disk buffers and drivers. Allocating three disk buffers leaves exactly 2048 bytes for resident drivers that will not be overwritten by the nonresident portion of DOS (contained in DUP.SYS).

IX. CUSTOMIZING A SYSTEM DISK

Custom RAMdisk Configurations

The RAMdisk driver included in MYDOS 4.3 is configured automatically for the Atari 130XE computer and uses its banked 64K bank of memory for the RAMdisk providing 499 free (single density, 128 byte) sectors. The 'O' command provides an easy way to alter the operation of the RAMdisk driver for other common banked memory systems. Most for the 800XL and 130XE use the same mapping address (the PORTB pins of the PIA chip in the computer). A 128K RAMdisk can be used in an Atari 130XE using the last unused pin of that port with no tradeoff (selecting the 64K bank is done with bit 6 of the byte written). If you have such a system, enter a '2' for the page sequence, and return for the others.

If, instead of adding one or two rows of 64K memory chips, the enhancement replaces the entire memory of the computer with a single bank of 256K memory chips, then the banked memory is a total of 192K and 4 bits of the port must be used to select the memory bank. Often the bits used are bits 0 and 1 (as in the 130XE) along with bit 6 (as in the expansion above) and bit 5 (used in the 130XE to control banking screen memory). Programs that bank screen memory (a very odd proposition because of the difficulty of obtaining a useful sharing of the banked memory page bits between the screen memory and the program) will not work with this enhancement.

This is the approach used in the Newell Industries 256K upgrade for the Atari 800XL.

If the enhancement is done externally or to an Atari 800 (with its internal expansion slots), a new dedicated register may be used to map the 16K pages. The Axlon RAMPOWER 128 card for the Atari 800 works this way. In such a system, the pages are selected by writing a page number to the mapping address and no sharing of the 8 bit byte is necessary. The address of the mapping register is entered explicitly and page sequence '5' is a proper sequence.

The page sequence tables coded into MYDOS 4.3 are actually one 64 byte sequence table with 3 posible permutations of the first 16 entries. A one digit sequence number specifies one of the following sequences:

Seq. No.	Page Values	OR Value
0	E3, E7, EB, EF, C3, C7, CB, CF, 83, 87, 8B, 8F, A3, A7, AB, AF	00
1	C3, C7, CB, CF, 83, 87, 8B, 8F, E3, E7, EB, EF, A3, A7, AB, AF	00
2	A3, A7, AB, AF, C3, C7, CB, CF, E3, E7, DB, DF, 83, 87, 8B, 8F	00
3 or 4	00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 0A, 0B, 0C, 0D, 0E, 0F	FF

As an example, you want to use BASIC/XE and you want to use pages E3, E7, EB, and EF from BASIC in the extended mode and pages 83, 87, 8B, 8F, C3, C7, CB and CF as a 128K RAMdisk. Selecting 128 as the RAMdisk size and entering either of the following lines --

1

or

C3,C7,CB,CF,83,87,8B,8F,0

sill configure the RAMdisk to use only that part of the banked memory not used by BASIC/XE.

The file RAMBOOT.AUT, the Atari AMAC assembler source code for which is in the file RAMBOOT.MAC, is an AUTORUN.SYS file that simulates the operation of Atari DOS 2.5 and its RAMdisk.

It "formats" the RAMdisk and copies DUP.SYS to it, then it sets the RAMdisk unit number and the unit used to access the DUP.SYS and MEM.SAV files to the unit selected for the RAMdisk.

By modifying the code in the source file and creating a modified AUTORUN.SYS file, the drive used to save MEM.SAV and fetch DUP.SYS can be modified, other files than just DUP.SYS can be copied to the RAMdisk when the system is booted or any other operation could be performed that you find useful.

Number of Files Open at Once

The number of files that may be simultaneously open is set with the same byte as in ATARI DOS 2: location $0709 (decimal 1801). This byte contains a number from 0 to 16 setting the number of disk files that may be open at the same time. Normally it is set to 3, the smallest number that supports all the functions in the MYDOS 4 menu. Specifically, a copy from one disk file to another requires three open disk files. The value in the distributed version of MYDOS 4 is three, to permit more or fewer files, use the "O" command followed by a RETURN.

To permanently change the maximum number of files, use the 'H' command to write a modified MYDOS 4 system to a disk.

Each file that may be open at one time requires the allocation of a 256 byte buffer so setting this value to 7 (instead of 3) will cause MYDOS 4 to be 1024 bytes longer than before and the programs loaded must begin no lower than $226C (instead of $1E6C). In corresponding fashion, by setting the value to 1, a BBS program can be loaded in with 512 bytes of additional memory if only one disk file is ever open at one time (commonly true of bulletin board programs).

Controlling the Disk Drives Accessed by MYDOS 4

Like ATARI DOS 2, MYDOS 4 automatically identifies the disk drives that are present when booted up initially and any time it is reinitialized (some programs do this on exiting to the DOS and it is always done if the RESET key is pressed. MYDOS 4 is distributed with drives 1 and 2 configured, all others are omitted in order to speed up the booting process. To modify the maximum configuration MYDOS 4 will use, invoke the 'O' command for each drive to be added to (or removed from) the system. Pressing the RESET key will then use this value to redefine the system. To permanently change the maximum drive configuration, use the 'H' command, writing a new copy of MYDOS 4 back to the system disk.

Selecting or Disabling Write-with-Verify

MYDOS 4 is distributed with all data written to the disk read back to verify that it was not only written to the diskette correctly, but that there was not a problem (dust, a scratch in the oxide coating, or some other problem that may have arisen since the diskette was formatted) that prevented the data from being read back from the diskette. If the programs being run have no long term value (games for example, often write daily high scores to the disk, and loss of such data might result in a few screams or moans, but so does waiting for a slow update of the scores after each game!).

The byte at location $0779 (1913 decimal) controls all write operations to the disk. If the value 'poked' into it is $57 (87 decimal), than all writes will be read back to verify the action was successful. If the value 'poked' into $0779 is $50 (80 decimal) then writes will be assumed successful, and will be performed in about one third the usual time. Note that this address is not the same as in MYDOS 4.0 and 4.1 (where it was $0770 or 1904 decimal).

This byte is defined, along with the count of the number of buffers to be allocated when the file manager is initialized, whenever the 'O' command is invoked with no drive specified (only a RETURN is entered in response to the drive number query). To permanently alter it, rewrite MYDOS back to the disk using the 'H' command after changing it.

The overhead associated with handling large RAMdisks and hard disks result in MYDOS 4 being somewhat slower than MYDOS 3 (and Atari DOS 2) when writing to the disk whether verify is enabled or not. The timing of disk I/O to permit maximum transfer speed is very critical and

MYDOS 4 is about 0.0015 seconds (per sector transfered) slower than MYDOS 3 (which takes about 0.1 seconds). With the standard sector interleave used by most Atari compatible drives, this adds another 0.2 seconds to each write operation.

X. DISK DRIVE INTERFACE (via SIO)

The physical disk drives and diskettes are external to the ATARI home computers and the ones supported by MYDOS 4 are normally attached to the "serial interface connector" on the right side or back of the computer.

High capacity or "hard" disks may also be connected to the parallel port of 800XL and 130XE computers. The software in the operating system (OS ROMs) to access the devices attached to either connector is call the "serial I/O driver" or SIO for short.

The MYDOS 4 disk operating system uses this lower level driver to pass all commands and information to and from the physical disk drive. Several commands were defined by ATARI to communicate with the 810 disk drive and most vendors of high performance disk systems for the Atari have adopted a slightly extended version of this set of commands. MYDOS 4 will operate in a limited fashion with any disk system that supports the entire original 810 set, but the full set of commands is required to support all the functions.

An additional function necessary to perform automatic density selection is that the drive automatically identify the density of a diskette inserted in it if the first operation is a read of sector 1 (this is necessary if the drive is to boot either a double or single density diskette).

The minimum set of disk drive functions to support MYDOS 4 (or ATARI DOS 2 for that matter) are:

Device	Unit	Command	Direction	Byte Ct.	Aux. Bytes	Function
$31	Drive#	$21	From Drive	128/256	1 to 720	FORMAT DISK
$31	Drive#	$50	To Drive	128/256	1 to 720	WRITE(no vfy)
$31	Drive#	$52	From Drive	128/256	1 to 720	READ
$31	Drive#	$53	From Drive	4	1 to 720	READ STATUS
$31	Drive#	$57	To Drive	128/256	1 to 720	WRITE(verify)

The byte count is always 128 for a small sector drive, and is 128 for the first three sectors (1, 2, and 3) of a large sector drive. All other sectors on a large sector drive are 256 bytes long. Standard double density diskettes have the large or 256-byte sectors and all others use the 128-byte sector size.

The FORMAT function is never called with a sector number not in the range of 4 to 720. It expects 128 bytes from a small sector drive and 256 bytes from a large sector (or double density) drive.

The first byte returned by the READ STATUS command is expected to indicate the sector size -- if bit 5 is a 1 (bit 7 is the sign bit) then the sectors are large (256 bytes), otherwise, they are small (128 bytes).

The auxiliary bytes are treated as an address to a sector on the diskette, and range from 1 to 720 (when in DOS 2 compatible mode) or from 1 to 65,535 (when accessing large capacity disk drives).

The additional functions used to configure disk drives dynamically are:

Device	Unit	Command	Direction	Byte Ct.	Aux. Bytes	Function
$31	Drive#	$4E	From Drive	12	1 to 720	READ CFG.
$31	Drive#	$4F	To Drive	12	1 to 720	WRITE CFG.

These commands are used to configure the drives identified as configurable when the computer is booted: if there is a possibility that a drive does not support these functions, it should be defined as not configurable (such as Atari and Indus drives). These commands also support the 'P' command, permitting reconfiguration of a disk drive on demand: to format a diskette, for example. (To format a disk on an Indus drive, issue the 'P' command, manually change the density on the drive, then issue the 'I' command).

The individual bytes transferred by these commands are defined as follows:

byte 0: Tracks per side (40 for a standard disk drive)
byte 1: Disk Drive Step Rate (as defined by Western Digital)
byte 2: Sectors/Track -- high byte (usually 0)
byte 3: Sectors/Track -- low byte(18 for standard diskettes)
byte 4: Side Code (0=single sided, 1=double sided)
byte 5: Disk Type Code --
bit 2: 0=single density, 1=double density
bit 1: 0=5 1/4 inch diskette, 1=8 inch diskette drive
byte 6: High byte of Bytes/Sector (0 for ATARI 810 compatible)
byte 7: Low byte of Bytes/Sector (128 for ATARI 810)
byte 8: Translation control
bit 7: 1=40 trk. disk I/O on an 80 trk. drive
bit 6: Always 1 (to indicate drive present)
bit 1: 1=Handle sectors 1, 2, and 3 as full size sectors
bit 0: 1=Sectors number 0-17 (for example) not 1-18
bytes 9-11 are not used by MYDOS (see the drive documentation as to how they are to be set -- usually zeroes)

MYDOS 4 (unlike earlier versions of MYDOS) always issues a read configuration command before writing the configuration to the drive and the contents of bytes 9-11 are written exactly as they were previously read (so they will be unchanged).

An additional change in the usage of this command occurs when a high capacity drive (hard disk) is configured. The configuration data for such a drive is very complex and is usually built into the drive controller or written to a "magic" location on the disk.

To support partitioning of very large drives (larger than 16 Megabytes) MYDOS issues a write configuration command with the number of sectors per track set to number of sectors on the disk (as defined in the 'O' command) and the number of tracks set to 1.

All high capacity disks are large sector drives (using 256 byte sectors).

XI. RAMDISK INTERFACE

The driver built into MYDOS 4 is intended to eliminate most of the need for a "driver" to use common RAMdisks. The required characteristics of the hardware can be most easily described by explaining what is done to access a "sector" of information in the extended RAM.

(1) the sector number is divided by 128, and the remainder is then multiplied by 128 and added to 16384 to get the starting address of the sector in memory (it will be between $4000 and $7F00).

(2) the quotient is used to index into a page table with one entry for each 16K that can be mapped into the memory area from $4000 to $7FFF.

(3) the value from the page table is "AND"ed with the contents of the mapping register and rewritten to the mapping register.

(4) the data is moved to(from) the area addressed above from(to) the sector buffers at the high end of MYDOS

(5) the mapping register is restored to its non-mapping state by "OR"ing the restore value with the mapping register and rewriting the result to the mapping register.

Note that this design forces the RAMdisk to be single density and no larger than 4 Megabytes (256 pages of 16384 bytes each).

As you can see, the parameters are the mapping register address ($CFFF for Axlon boards and $D301 for the Atari 130XE), the value "OR"ed into the register to reset the system back to normal (usually $FF for the Axlon and $00 for the Atari 130XE), and the actual map values. These values are determined by first identifying the bits in the mapping register to be left unchanged and setting them to "1" in each of the register values.

Second, the remaining bits are filled in with all the legal combinations of mapping bits. The values for the Newell Industries 256K upgrade (which uses the 130XE mapping, more or less) are given here as an example -- future versions of this board and other memory expansion products are not necessarily going to use the same design.

Bits: 7 6 5 4 3 2 1 0

1	x	x	x	x	x	1	1	First, set bits 7, 1 & 0 in all the mapping values
1	0	0	0	0	0	1	1	These are the 12 (of 32)
1	0	0	0	0	1	1	1	
1	0	0	0	1	0	1	1	
1	0	0	0	1	1	1	1	
1	1	0	0	0	0	1	1	
1	1	0	0	0	1	1	1	
1	1	0	0	1	0	1	1	
1	1	0	0	1	1	1	1	
1	1	1	0	0	0	1	1	
1	1	1	0	0	1	1	1	
1	1	1	0	1	0	1	1	
1	1	1	0	1	1	1	1	

Lastly, since the mapping register at $D301 can be read as well as written, it can be left exactly as it was before we used it by "OR"ing the initial value with zero (leaving it unchanged). The sequence is then: 83, 87, 8B, 8F, C3, C7, CB, CF, E3, E7, EB, EF, 0.

XII. INITIAL INSTALLATION INSTRUCTIONS

The following checklist is a procedure to bring up MYDOS 4 on a new hardware configuration. It should be done with an unmodified copy of the MYDOS 4.3 distribution disk and care should be taken to perform the steps in exactly the order specified.

HOW TO GET FROM THE DRIVEWAY TO THE FREEWAY:

1. Before turning on your computer and disk drive, make sure that you do not have a cartridge installed in the computer.

2. Turn on the disk drive, and when it stops spinning, insert the MYDOS master diskette. Close the door if it has one.

3. Turn on the computer: after several seconds, the MYDOS 4 menu should appear on the screen. A prompt 'Select Item (RETURN for menu):' will be displayed.

4. If you have a standard 2-drive system with single sided 40 track drives and either no RAMdisk or the standard Atari 130XE memory configuration continue with step '6'.

5. Configure each of your disk drives by entering the 'O' command followed by the drive number, and answering the questions the program asks you.*

6. If you need to run a program that requires more than 3 disk files active at a time or you otherwise want alter one of the default parameters, press the 'O' key followed by the RETURN key and answer the system configuration questions. This is also

where the RAMdisk configuration needs to be entered if it is not a standard 130XE RAMdisk or no RAMdisk exists in your system. ****

7. Now, insert a blank diskette in the disk drive and format it with the 'I' command (remove the write protect tab if the diskette is write protected -- but first ask yourself why you put it there in the first place!). If an error message is displayed, insert another diskette, the first may be usable but it is not a good choice for your system disk or system disk backup.

8. Write the system files DOS.SYS and DUP.SYS to the newly formatted diskette with the 'H' command and skip to step '10' if you do not have an ATARI 130XE computer or some other RAMdisk usable with the MYDOS RAMdisk driver.

9. If you have an ATARI 130XE computer or have another kind of RAMdisk and wish to use the MYDOS RAMdisk driver, copy RAMBOOT3.AUT on the master diskette to AUTORUN.SYS on the new diskette (use the 'C' command, and if you have only one drive, enter the command line 'RAMBOOT.AUT,AUTORUN.SYS/X' when MYDOS asks for the file names).**

10. This is now your primary system backup disk: place a write protect tab on it, label it and use the 'J' command to make as many working boot disks as you need from it.

NOTES:

The RAMBOOT3 program will copy the files in a directory called RAMDISK on your boot disk to the RAMdisk if such a directory exists. To take advantage of this feature, use the 'Q' command to create the directory and copy the files you want on the RAMdisk to it.

* -- The step rate is a cryptic code that ranges from 0 (fast) to 3 (very slow). Most drives will work with 0 or 1 but you may need to experiment with it to find the best speed for your drives (and power supplies).

** -- The source file for the RAM disk boot is also included, if you feel the urge, modify it perform other tasks -- RAMBOOT2 configures the RAMdisk, formats it, moves it to drive 8, enables MEM.SAV and copies DUP.SYS to it, RAMBOOT3 configures it, formats it, enables MEM.SAV, copies DUP.SYS to it and also copies all the files in D1:RAMDISK: to it. RAMBOOT3, the one distributed as AUTORUN.SYS must also be the last AUTORUN file in AUTORUN.SYS since it uses all 3 disk buffers in a standard MYDOS. An improved version without this limitation will be release with the next version. If you recompile RAMBOOT2 and use it, remember that when you string togather several AUTORUN files, only the last will have its RUN address invoked.

**** -- NEWELL INDUSTRIES 800XL UPGRADE --

this RAMdisk can use page sequence 0 or page sequence 1. The E3/E7/EB/EF pages can be reserved for BASIC/XE in extended mode by setting the RAMdisk size to 128K and the sequence to 1. Enter Size=128 (or 192 if you will not be using BASIC/XE often), Control address=D301, and Page sequence=1. In the second case this will use the area also used by BASIC/XE, but it will access it only after the first 128K are used up. By setting the size to 128K, BASIC/XE and the RAMdisk are fully separate and cannot interfere with each other.

**** -- AXLON RAMPOWER 128 for the Atari 800 computer --
MYDOS 4.0 comes configured for the Atari 130XE -- if you are installing it on an 800 with an Axlon RAMPOWER board, you must disable the 128K mode on the RAMPOWER board until you have configured the MYDOS RAMdisk code by entering Size=112, Control address=CFFF and Page sequence=5.
Then write the DOS back to a new diskette, copy AUTORUN.SYS to it, switch the RAMPOWER board back to 128K and boot up the new diskette.

XII. ERROR CODES AND THEIR SOURCES

3	Last byte of file read, next read will return EOF (MYDOS)
128	Break Abort (OS ROMs)
129	IOCB already open (OS ROMs)
130	No such device defined in the system (OS ROMs)
131	Write-only IOCB, cannot read (OS ROMs)
132	Invalid command (OS ROMs)
133	Device or File not open (OS ROMs)
134	Invalid IOCB reference (OS ROMs)
135	Read-only IOCB, cannot write (OS ROMs)
136	Attempt to read past end of file (MYDOS)
137	Truncated record (OS ROMs)
138	Device Timeout (OS ROMs)
139	Device NAK (serial bus failure, OS ROMs)
141	Cursor out of range for graphics mode (OS ROMs)
142	Data frame overrun (serial bus failure, OS ROMs)
143	Data frame checksum error (serial bus failure, OS ROMs)
144	Device I/O error (in peripheral hardware, OS ROMs)
146	Function not provided by handler (OS ROMs)
147	Insufficient RAM for graphics mode selected (OS ROMs)
160	Invalid Unit/Drive Number, zero or greater than 9 (both MYDOS and OS ROMs)
161	No sector buffer available, too many open files (MYDOS)
162	Disk full, cannot allocate space for output file (MYDOS)
163	Write protected or system error - if the disk is not write protected, it should be copied to another disk immediately -- the file system is corrupted (MYDOS)
164	File number in link does not match the file's directory location (MYDOS)
165	Invalid file name (MYDOS)
166	Byte not within file, invalid POINT request (MYDOS)
167	File locked, cannot be altered (MYDOS)
168	Invalid IOCB (MYDOS and OS ROMs)
169	Directory full, cannot create a 65-th entry in a directory -- entries may be used for 'lost' as well as real files (MYDOS)
170	File not in directory, cannot be opened for input (MYDOS)
171	IOCB not open (MYDOS and OS ROMs)
172*	File or directory of same name already exists in parent directory, cannot create (MYDOS)
173	Bad diskette or drive, cannot format diskette (MYDOS)
174*	Directory not in parent directory (MYDOS)
175*	Directory not empty, cannot delete (MYDOS)
180*	Invalid file structure for loading memory (MYDOS)
181*	Invalid address range for loading memory, END<BEGIN (MYDOS)

* -- New error codes, not present or different in Atari DOS 2.

Most error codes are identical to those returned from ATARI DOS 2, the differences result from the expanded capabilities of MYDOS 4. Specifically, Error 164, indicating a file number mismatch, only occurs if the file is written in DOS 2 or ATARI DOS I format. Errors 180 and 181 can only occur when XIO 39 is invoked to load a file, the ATARI DOS 2 equivalent function returned a code in the X-register. Errors 172 and

175 apply to creating and deleting directories and have no ATARI DOS 2 equivalent. And Error 174 applies to accessing files in subdirectories, so it also has no ATARI DOS 2 equivalent. Error code 173 serves the same function as it did in ATARI DOS 2, but is returned more often (to identify bad diskettes more reliably).