The MYARC 9640 Family Computer

The MYARC 9640
MDOS and **GPL**

Revised 1/22/00
Typeset 10/16/06

TABLE OF CONTENTS

INTRODUCTION

In the years since Texas Instruments left the home computer market there has been great speculation about the appearance of a remarkable, next generation computer. Using the latest technology we have created that new generation, the MYARC 9640 Family Computer.

The speed, memory, graphics, and processing power of the MYARC 9640 Family Computer surpasses that of all other microcomputers of its class. Since the 9640 is the next generation stemming from the Texas Instruments 99/4A home computer, it is designed to be compatible with the many thousands of programs presently available for the TI-99/4A.

With over 512,000 bytes of random access memory, expandable to over 2,000,000 bytes of RAM, the 9640 has the capacity to facilitate the most complex word processing tasks or most detailed spreadsheet procedures. Application programs of all kinds take on new potential when they are written to take advantage of the graphics and speed of the 9640 computer.

The MYARC 9640 computer offers you all the computing power you need: 512 different colors, seven graphics modes, both 40 and 80 columns of text, special graphic capabilities, and awesome processing speed. And with the ability to add hard (fixed) disk systems, modems, and printers, the 9640 computer can grow with your needs and adapt to new uses.

ABOUT THIS MANUAL

This manual, "The MYARC 9640", introduces you to the MYARC 9640 Computer. It can be used to learn the basic skills needed to operate the computer, and can also serve as a handy reference guide.

This manual will:

- Show you how to set up and operate the 9640 computer.
- Describe and teach you MYARC DOS. MDOS (disk operating system) is the link between you and the computer.

OTHER MYARC MANUALS

"MYARC Advanced BASIC" describes and teaches MYARC Advanced BASIC which is the program language included with the 9640.

"Technical Reference" provides an overview and technical description (with reference information) on all operating systems in the 9640, their interrelationships and interdependences.

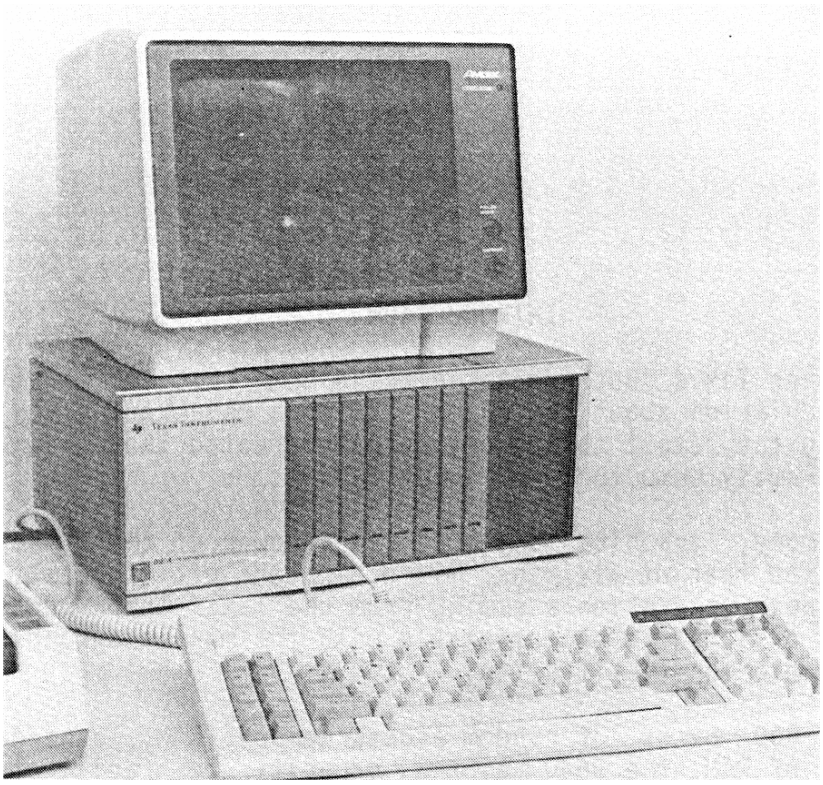

Figure 1 – 9640 Typical System Setup

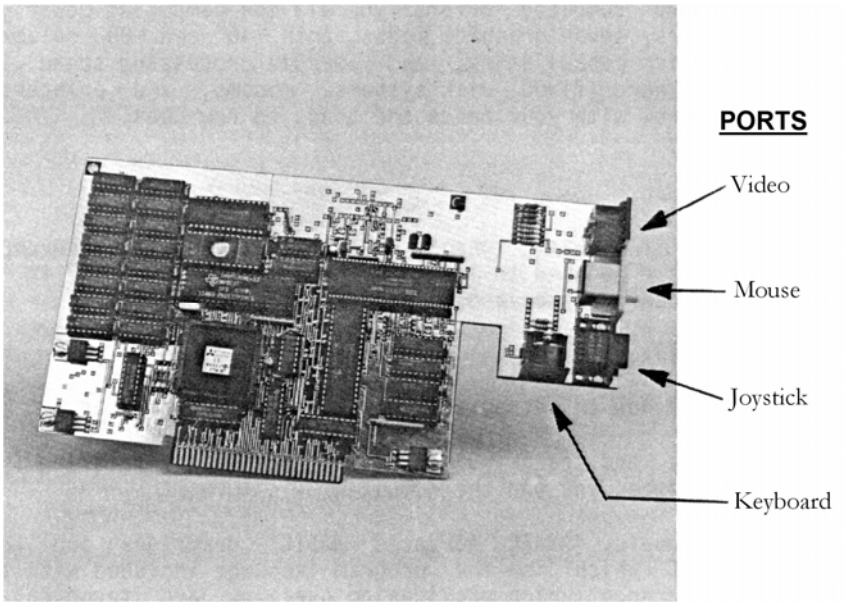

Figure 2 – The 9640 System Board (removed from clamshell case). Ports are at the rear of the board.

MEET THE MYARC 9640 COMPUTER

The MYARC 9640 computer includes two major components; a detachable keyboard, and a (peripheral expansion) 9640 Computer Card containing the MYARC system board. When the system board is installed into a peripheral expansion system, the components are transformed into the most powerful and sophisticated computer ever offered in the family and small business area.

To use the MYARC 9640 computer you will need a monitor and a floppy (or fixed) disk system. These items, along with a variety of optional expansion equipment, are available from MYARC and other manufacturers. Figure 1 illustrates a typical 9640 system configuration showing the TI Peripheral Expansion Box, two disk drives, a monitor, and the keyboard (the system board, or computer, is located inside the peripheral expansion system).

THE KEYBOARD

The 101-key full stroke keyboard (See Figure 5, page 14) contains all the keys found on a conventional typewriter, as well as special (gray) keys that will assist you in a variety of tasks. At each end of the keyboard there is an adjustable leg that can be used to place the unit in either a level or tilted upright position. The keyboard attaches to the rear of the system board via a coiled, expandable cable.

THE SYSTEM BOARD

The heart of the MYARC 9640 computer is its system board. Using the latest technology, this small 7 by 5 inch board contains the following:

Texas Instruments TMS9995 12MHz microprocessor

- 512K bytes (K = kilo = 1000) of user addressable random access memory (RAM)
- Complete high resolution video display processing system which includes a Yamaha V9938 Video Display processor (VDP), 128K VDP RAM, RGB analog video port, and composite video port
- SN76496N sound processor with the 3 simultaneous voices and 1 noise generator
- Battery-backed real time clock
- MYARC mouse port
- Joystick port

Figure 2 shows a top view of the 9640 System Board. Note the locations of the video, mouse, joystick, and keyboard ports at the back (to the right in the photo) of the computer board.

PERIPHERAL EXPANSION CARDS

A variety of peripheral expansion cards, or boards, are available from MYARC and other manufacturers which provide the ability to further enhance the capabilities of the 9640 computer.

It is recommended that you check with your local dealer, or MYARC, before inserting present expansion cards into the peripheral expansion system containing the 9640 computer. Incompatibilities may arise from the use of cards originally intended for the TI-99/4A computer.

MEMORY EXPANSION CARDS

MYARC 128K, 256K, and 512K cards can be used to expand your computer's memory significantly. Using a single 512K memory expansion card, you could boost your available addressable memory to 1 megabyte.

Only those memory expansion cards using the MYARC 9640 memory specification can be used to expand your computer's memory. All existing 32K memory expansion cards, and other larger memory expansion cards, originally intended for use with the TI-99/4A computer cannot be used with the 9640 computer.

CONVERTING MYARC MEMORY EXPANSION CARDS FOR USE

Present MYARC 128K, 256K, and 512K memory expansion cards originally manufactured for use with the TI-99/4A must be converted so they may be used with the 9640 computer.

DISK CONTROLLER CARDS

Using a disk controller card, disk drives can be attached to the system that will allow you to easily store and retrieve information from 5¼ or 3½ inch diskettes.

Three different disk controller cards are presently available that will operate with the 9640 computer.

HARD (FIXED) DISK CONTROLLER CARD

Using the MYARC HFDC hard disk controller card, three hard disks can be attached to the system that will allow you to store and retrieve an enormous amount of information much quicker than you possibly could with floppy disk drives. The MYARC HFDC can utilize most ST-506 compatible hard disk drives with a total maximum storage capacity of 402 (3x134) megabytes.

RS232 INTERFACE CARDS

Most RS232 interface cards combine both serial and parallel interfaces ("ports") into one expansion card that can be used to transfer data both to a serial and/or a parallel device.

Many serial interfaced devices, such as a modem or serial printer, can be attached to the system via an RS232 port. The parallel port is usually used to attach a parallel printer to the system. Up to two RS232 interface cards can be installed inside the peripheral expansion system at the same time (a slight modification to the second RS232 interface is required.)

Three different RS232 interface cards are presently available that will operate with the 9640 computer.

MONITORS

Two ports, a composite video port and an RGB analog video port, are available for attaching a monitor to the 9640 computer. Each of the two ports has its advantages as well as its disadvantages.

COMPOSITE VIDEO PORT

The composite video port (see Figure 2) allows you to connect a composite monitor, or television set with an RF modulator, to the 9640 computer.

Generally, standard composite MONOCHROME monitors will display both 40 and 80 columns of text and high resolution graphics adequately. However, most composite COLOR monitors and television sets can only display satisfactorily 40 columns of text and low resolution graphics.

RGB ANALOG VIDEO PORT

The RGB (red, green, blue) analog video port (see Figure 2) allows you to connect a high resolution analog RGB color monitor to the 9640 computer. Monitors of this type will display both 40 and 80 columns of text and high resolution graphics with extreme clarity in full color.

SETTING UP THE 9640 SYSTEM

For most systems, assembling the components of your 9640 computer system in a work area four to six feet from a dual power receptacle will provide adequate access to power. If many peripherals that are independently powered (such as a modem or a printer) are to be attached to the system, additional power outlets will be needed. A power strip can be connected to your household receptacle to obtain additional outlets, or you can use an extension cord. A standard six-plug power strip is recommended, since most include circuit breaker protection against overloads and have a built-in reset button, making it a safer, neater alternative to dangling extension cords.

UNPACKING THE SYSTEM

Carefully unpack the 9640 computer system board, its keyboard, and the accompanying accessories. It is a good idea to save the packing materials for storing and/or transporting your computer system.

At this stage you will only need the following:

- 9640 Computer System Board (Card)
- 9640 Keyboard

HOOKING UP THE SYSTEM

Setting up you 9640 computer system is easy. Since all the devices simply plug together, or into your TI Peripheral Expansion Box, all you need do is to plug each device into the right place.

The peripheral expansion system (PE box) has eight slots into which peripheral expansion cards can be inserted. Assuming that you have been using the Texas Instruments peripheral expansion system with a TI-99/4A computer, you MUST remove the following equipment from the expansion system:

- Peripheral Expansion Card (located in slot 1.)
- Memory expansion card. This includes all 32K cards, and any other larger memory expansion cards originally used with the TI-99/4A.
- Any other peripheral expansion cards that may cause incompatibility problems (check with your local dealer or MYARC, for additional information.)

Carefully, following the step-by-step instructions below, insert the 9640 system board into the peripheral expansion system and attach any peripheral devices.

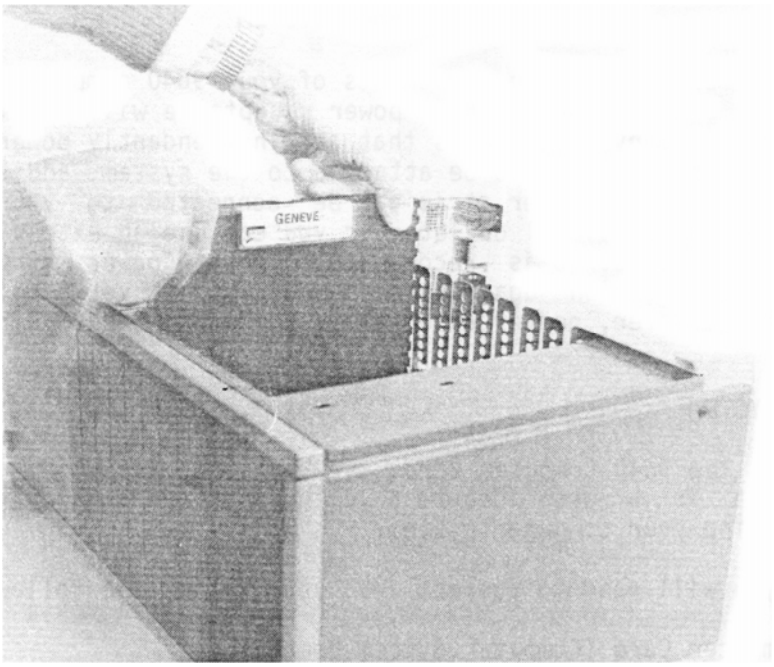

Figure 3 – This is the 9640 computer card. Connector sockets are at the rear

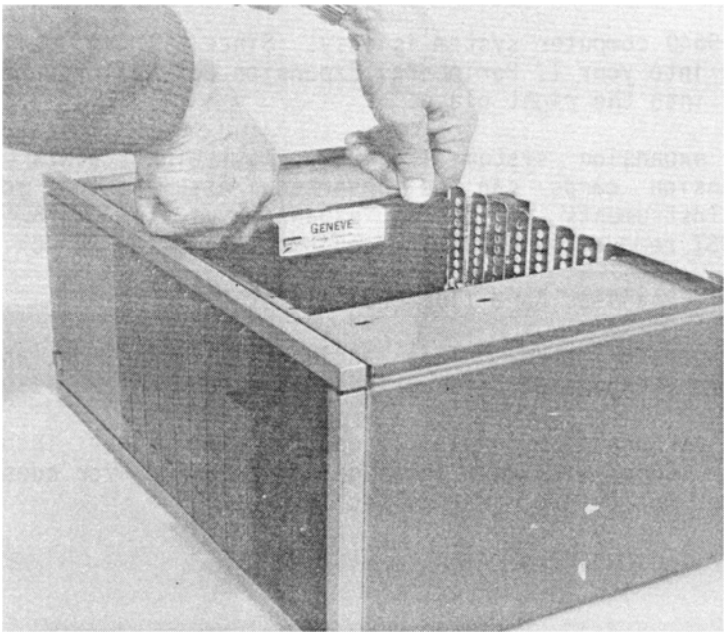

Figure 4 – Inserting the 9640 computer card into the Peripheral Expansion Box

1. Make sure the peripheral expansion system is turned off.

2. Remove the top of the peripheral expansion system by lifting the back edges of the top and pulling up.

3. Remove any peripheral expansion cards that cannot be used with the 9640 computer. Warning: To avoid damaging any of the peripheral expansion cards, wait two (2) minutes after turning the expansion system off before removing any expansion cards.

4. FOLLOW THIS STEP ONLY IF YOU ARE HOOKING UP AN ANALOG RGB MONITOR.

The combination video port is factory preset for using an RF modulator interface (monochrome monitor, composite color monitor, or television set) and, therefore, you will need to reset the combination port for your analog RGB color monitor.

Before inserting the MYARC 9640 Computer Card as in step 5 below, reset the pin located on the board directly in back of the combination video port, from GND to R (RED). See Figure 2, page 7. Also refer to "Connector Pinouts" (Appendix E).

5. The label identifying the 9640 system board is on the very top of the card. At the rear of the Card are the keyboard, video, mouse, and joystick connector ports. Hold the 9640 Computer Card (label side up) with the connector ports facing towards the rear of the PE box. See Figure 3.

6. Carefully align the 9640 Computer Card in the first slot (marked by a 1) and press the card firmly down into the slot. You should be able to feel the connection being made between the 9640 Computer Card and the PE box. See Figure 4.

7. Replace the top on the PE box.

8. Insert your keyboard, monitor, mouse, and joystick cables into the proper ports at the rear of the 9640 Computer Card. See Figure 2, page 7.

ADDING ADDITIONAL PERIPHERAL EXPANSION CARDS

When installing additional expansion cards into the peripheral expansion system, follow the instructions furnished with that expansion card. The instructions should explain how to insert the expansion card into the expansion system, and how to attach any associated devices.

If you have any questions about installing or operating additional peripheral expansion cards, please contact your local dealer, or MYARC.

REMOVING PERIPHERAL EXPANSION CARDS

The following instructions explain how to remove a peripheral expansion card from the peripheral expansion system.

1. Make sure the expansion system is turned off.

2. Remove the top of the expansion system by lifting the back edges of the top and pulling up.

3. Disconnect any cables that may be attached to the expansion card you want to remove.

4. Use both hands to pull up on the expansion card to remove it from its slot in the expansion system. Warning: To avoid damaging any of the peripheral expansion cards, wait two (2) minutes after turning the expansion system off before removing any expansion cards.

TURNING ON THE SYSTEM

You should always turn on your computer and peripherals in the following sequence:

1. External peripherals (such as a disk drive, printer, or modem).

2. Monitor or television set.

3. Peripheral expansion system (which houses the 9640 system board, various expansion cards, and internal disk drives.)

The process of preparing your computer for use, which includes loading the disk operating system, is fully explained in "Starting MDOS".

UNDERSTANDING THE KEYBOARD

KEYBOARD LAYOUT

The computer's keyboard (Which is a XT style) contains 101 keys including the space bar, and is divided into three sections as partially illustrated in Figure 5. The top row is comprised of 12 function keys, while the typewriter key area is in the left section. A numeric keypad is in the third section on the right-hand side of the keyboard. On the following keys are status lights that display the status of the Capitals (Caps) Lock, Numeric (Num) Lock, and Scroll lock.

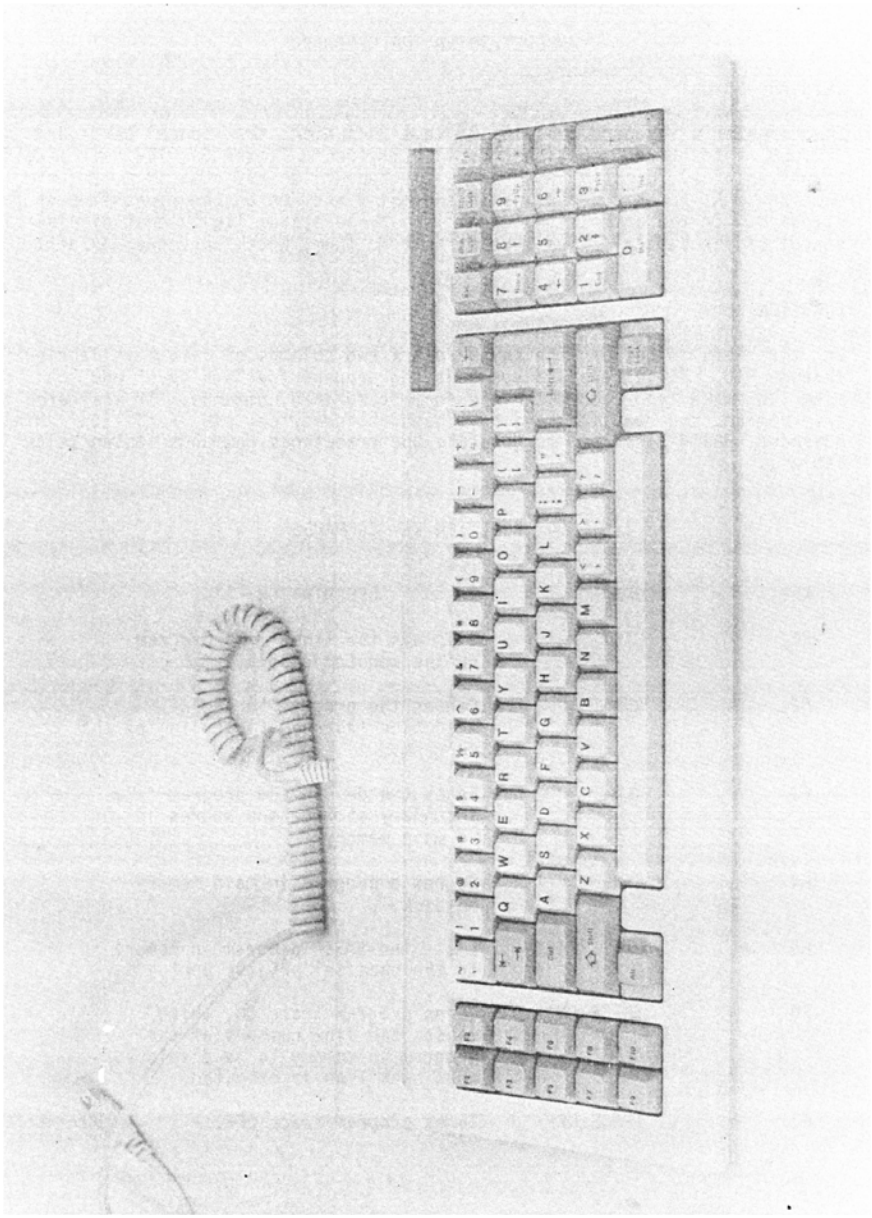

Figure 5 – 9640 keyboard

FUNCTION KEYS

On the top row of the keyboard there are 12 keys labeled F1 through F12. These keys are known as the program function keys and can be used to make your computer perform predefined commands in various programs.

PROGRAMMING ASSISTANCE KEYS

Surrounding the normal typewriter keys in the middle section of the keyboard is a series of keys that can be used for assistance in writing, updating, and executing programs. These keys are listed below in Table 2 along with a description of their operation.

Table 2. Programming Assistance Keys (in MDOS)

KEY	MEANING	DESCRIPTION
Tab	Tab	Performs a tab function similar to a typewriter. The shift key reverses direction.
Ctrl	Control	Always used with a second or third key to perform a function or command.
Shift	Shift	Changes lowercase to capitals or if Caps Lock is on changes capitals to lowercase letters.
Alt	Alternate	Used for supplemental command entry.
Backspace	Backspace	Moves the cursor to the left same left arrow
Enter	Enter	Indicates the logical end of a line of input by moving the cursor from the last character on one line to the first character of the next.
Space Bar	Space	Moves the cursor one position to the right.
Caps Lock	Capitals Lock	A toggle key that causes letters to be typed in upper case when on (indicated by the status light), lowercase when off.

NUMERIC KEYPAD

When the Num Lock key is pressed, the numeric keypad will become activated. This will cause the cursor keys to switch to number keys. The keypad is arranged like a calculator and permits easy entry of large quantities of numeric data.

With the Num Lock key activated, keys 1 through 9 on the numeric keypad produce the digits 1 through 9, while the Del key produces a decimal point and the Ins key results in a zero.

CURSOR CONTROL AND EDITING

When the Num Lock key is not activated, the numeric keypad keys take on alternate meanings and are used for cursor control and screen editing the same as the center section dedicated cursor control keys. The meanings of the dedicated keys and the alternate meanings of the numeric keypad are indicated below in Table 3.

Table 3. Cursor Control Key Pad Functions

KEY	DESCRIPTION
Home	Repositions the cursor to the first character of the top line of the screen.
Up Arrow	Moves the cursor up one line for each keystroke.
Left Arrow	Moves the cursor to the left one character position for each keystroke.
Right Arrow	Moves the cursor to the right one character position for each keystroke.
Down Arrow	Moves the cursor down one line.
End	Positions the cursor at the last character of the current line.
Del	Deletes the character where the cursor is positioned.
Ins	Sets the keyboard to the insert mode of operation; other characters keyed are displayed to the right of the cursor and all data already on that line will move to the right. Terminate the insert mode by pressing the Ins key again or press the Enter key if all line modification has been completed.
Pg Up	Scrolls screen one screen up.
Pg Dn	Scrolls screen one screen down.

The Escape (Esc) key will perform functions defined in your operating system or by the application program being used. Normally, this key is used to remove the line that the cursor is on for corrections.

When the Scroll Lock key is pressed, the Scroll Lock light will be illuminated. There is no present use for this.

The Print Screen (PrtSc) key will cause the data that is displayed on your screen will be printed.

MULTI-KEY OPERATIONAL COMBINATIONS

The depression of two or three keys simultaneously can be employed to perform a series of unique program and control functions. These functions are listed in Table 4. Additional multi-key operational combinations applicable only to the use of the Disk Operating System will be described later.

Table 4. Multi-Key Program and Control Keys

KEY	FUNCTION	DESCRIPTION
Ctrl+Break Ctrl+C	Break	Causes the execution of a program to terminate and identifies the line where it stops.
Ctrl+Num Lock Pause	Pause	Suspends program execution; press any key to continue program execution.
Ctrl+Tab	Word Tab	Moves the cursor forward to the next word on the current line.
Ctrl+Home	Clear Screen	Clears the entire screen and moves cursor to the first character position on the first line.
Ctrl+Alt+Del	System Reset	Causes the system to reset and load the Disk Operating System.
Shift+PrtSc	Print Screen	Causes all data on the screen to be printed

THE DISK OPERATING SYSTEM

WHAT IS MDOS?

The MYARC Disk Operating System (DOS) is a collection of programs that gives you complete control over what your computer does and how it does it. Without MDOS you would probably be unable to use application programs such as a word processor or a spreadsheet. MDOS is the link between you and your computer.

Your computer equipment, called hardware, probably includes a keyboard, monitor, printer, and one or more disk drives. No matter how powerful the hardware, a computer cannot do anything without programs, called software. There are two major types of software: system programs, which control the operation of the computer system, and application programs, which perform tasks such as word processing.

Different application programs may use the hardware to perform similar tasks in different ways. These tasks could include receiving instructions from the keyboard, displaying information, printing information, reading and writing files to and from a disk, sending and receiving data through a communications port, and so on through all the capabilities of the hardware. Rather than have each program perform all of these functions for itself, a system program called the operating system manages the hardware.

Since much of the work performed by the operating system involves managing disks and disk files, it is referred to as a disk operating system, or DOS. MDOS coordinates all of the functions of your computer and controls the things you care about, such as which program to run, what report to print, or what files to erase.

WHAT CAN YOU DO WITH MDOS?

MDOS coordinates the operation of the computer for your application programs, but MDOS also has much more to offer. You can use MDOS, controlling it with instructions called commands, to manage your files, control your work-flow, and perform useful tasks that might otherwise require additional software.

You can tailor MDOS to meet your specific needs by creating powerful commands made up of other MDOS commands, and you can even create your own applications.

DISKETTES

The MYARC 9640 computer can use 5¼ and 3½-inch diskettes (floppies) and fixed disks for storing information.

TRACKS, BYTES, and SECTORS

Information in the form of data or programs is written onto and read from the diskette along concentric circles called tracks. There are 40 tracks on a conventional diskette that are numbered from 0 to 39, while a high-capacity diskette contains 80 tracks, numbered from 0 to 79. Depending on the type of disk drive and disk drive controller used to format the diskette and the FORMAT command specification you use, each track can be subdivided into 9, 16, 18 or 36 sectors. Each sector can store up to 256 bytes of information, where a byte represents 8 bits or one character of data. The FORMAT command prepares a diskette to receive information, checks the diskette for bad sectors, and builds an empty directory that will eventually hold information about the files that will be written onto it. This command, as well as other operating system commands, will be covered later.

DISK DRIVES

The MYARC 9640 computer is capable of using two types of disk drive systems: a floppy disk drive system that uses removable flexible diskettes, and a fixed disk system that makes use of non-removable media.

Depending on the type of floppy disk controller your system contains, your computer may use the following types of disk drives with their respective storage capacity.

- Single-sided, single-density (90KB)
- Single-sided, double-density (160KB/180KB)
- Double-sided, single-density (180KB)
- Double-sided, double-density (320KB/360KB)
- Double-sided, quad-density (640KB/720KB)
- Double-sided, high-density (1.44MB)

A single-sided single-density diskette contains 40 tracks, 9 sectors per track, and holds up to 90K bytes of information.

A single-sided double-density diskette contains 40 tracks, 16/18 sectors per track, and holds up to 160K/180K bytes of information.

A double-sided single-density diskette contains 40 tracks per side, 9 sectors per track, and holds up to 180K bytes of information.

A double-sided double-density diskette contains 40 tracks per side, 16/18 sectors per track, and holds up to 320K/360K bytes of information.

A double-sided quad-density diskette contains 80 tracks per side, 16/18 sectors per track, and holds up to 640K/720K bytes of information.

A double-sided high density diskette contains 80 tracks per side, 36 sectors per track, and holds 1.44M bytes of information.

Some combinations for reading and writing between different diskette and drive types are not allowed. The following describes which diskette and drive combinations are allowed.

- Single-sided drives can read and write to single-sided diskettes.
- Double-sided drives can read and write to both single-sided and double-sided diskettes.
- Single-density drives can read and write to single-density diskettes.
- Double-density drives can read and write to both single-density and double-density diskettes.
- Quad-density drives can read and write to single-density, double-density and quad-density diskettes.

Restrictions also apply as to what types of diskettes each particular disk controller may read and write to. The following describes which diskette format each disk controller is capable of using.

- Texas Instruments PHP1240 Controller: single-sided, double-sided, and single-density diskettes.
- MYARC DDCC-1 Controller: single-sided, double-sided, single-density, and double-density (both 16 and 18 sector formats). Quad-density capabilities are also available with the installation of the custom MYARC quad-density FDC-80 upgrade chip.
- CorComp 9900 Controller: single-sided, double-sided, single-density, and double-density (18 sector format only.)
- MYARC HFDC can write to all formats in addition to hard disks.

DISK FILES

Just as you organize and store your written records in paper files, you organize and store computer information in disk files. A disk file, or simply a file, is a collection of related information stored on a disk. It could be a letter, a list of customers, or even a program. Virtually all your computer work will revolve around files.

CARE AND HANDLING OF DISKETTES

You should always be extremely careful when handling and storing your diskettes to prevent accidental loss of data and programs. If possible, always consider the following factors with respect to disk handling and storage.

- Never touch the exposed recording surfaces of the diskette (the large oval area at the bottom of the disk where data is actually read and the small index hole near the center ring of the disk).
- Due to the fragile nature of diskettes, always try to store them in an upright position in their envelope to ensure that they do not bend or sag.
- Never place heavy objects on top of your diskettes.

- As soon as you remove your diskette from the drive, place it in its envelope to prevent the accumulation of dust or fingerprints on its head slot.
- Store your diskettes in appropriate storage boxes away from sunlight and other heat sources as well as magnetic field sources such as telephones, electronic calculators, and other electronic equipment.
- If you label the information on your diskette, do so only with a felt tip pen to avoid damaging the diskette. Write on the labels, whenever possible, before putting them on the diskette.

WRITE PROTECTION

The write protect notch provides a means to safeguard information recorded on your diskette from accidental erasure. To write-protect a 5¼-inch diskette, cover the notch with a piece of non-transparent tape, usually referred to as a write protect tab. If you later wish to write additional information or change old information on the diskette, simply remove the write-protect tab. To write-protect a 3½-inch diskette, slide the write protect notch (upper left corner of the disk, if you are looking at the disk with the label side at the top and facing you) so that you can see through the hole. To write to the disk, slide the notch back so that it blocks the hole.

COMMAND SYNTAX (FORMAT)

A common format notation will be used in discussing each of the MDOS commands.

- Words in capital letters are keywords. The specific characters in the keyword must be entered, although any combination of upper and lowercase characters can be used.
- The items shown in lowercase letters are to be supplied by you when you enter the command.
- All items in square brackets ([ ]) are optional and may or may not be included in a command.
- Items that may be repeated as many times as you wish are indicated by ellipses (...).
- With the exception of square brackets, all punctuation characters such as commas, equal signs, slashes, colons, question marks, and backslashes, must be included as by the command format.

COMMAND PARAMETERS

Parameters are items that you can include in your MDOS command statements. They are used to specify additional information to the system. Some parameters are required in your commands while others are optional. If you do not include some of the command parameters, the system will provide a default value. The following command parameters and notation will be used with each command:

d: - Denotes when you should specify a disk drive letter. Enter a drive letter followed by a colon to specify the drive. If you omit this parameter MDOS assumes the current default drive.

Filename - Diskette file names can be up to ten characters in length. Paths will be used with fixed disks and tree structured directories. The use of pathnames in commands will be discussed in a later section "Fixed Disk Organization".

STARTING MDOS

Whenever you start your computer, whether it is to use an application program, or MDOS itself, you will begin by loading MDOS into the computer memory. Loading the MDOS program and starting it running is sometimes called "booting the system" or "booting the disk".

Although most systems only consist of floppy disk drives, special attention will be given to those systems that also have a fixed disk installed. If you are using a fixed disk, the examples assume you have prepared the fixed disk so MDOS may use it. If you haven't yet prepared the fixed disk it is suggested that you do so now.

STARTING THE SYSTEM

If you're not using a fixed disk, open the latch of drive A (the first logical drive of the system) and put in the MDOS system diskette. When the disk is fully inserted close the latch.

If your computer is off, turn on the system in the proper manner. The computer will seem to do nothing for a few seconds, but this is normal. Each time you power the system up, the computer checks its memory and all attached devices to be sure everything is working properly. If you are not using a fixed disk, MDOS is copied into the system's memory from the diskette in drive A. If you are using a fixed disk with MDOS on it, the MDOS program is copied into the system's memory from the fixed disk (through LOAD/SYS). As soon as MDOS is loaded it will be ready to go to work. This process is usually referred to as a "cold boot". Finally, once MDOS is loaded, it will look for a text file called "AUTOEXEC" and, if it's present, MDOS will execute the commands it finds in that file.

SETTING THE TIME

The first thing you see after MDOS is loaded and starts running is the following message:

```
Current time is 00:01:30
Enter new time:_
```

The blinking underline that follows the colon is the cursor. It shows where MDOS will display whatever you type next. It also tells you that MDOS is waiting for you to type something--in this case, the time in response to its request. Such a request is called a prompt.

To set the time to 10:45 you would type and enter the following:

```
Current time is 00:01:30
Enter new time: 10:45
```

MDOS works on the basis of a 24-hour clock. For example, if you wanted the time to be set to 3:45 p.m., you would enter it as 15:45. If you were to enter 3:45 p.m. as 3:45, the system time would actually be 3:45 a.m.

The time is entered by typing in the numbers that represent the current hour and minute separated by a colon. MDOS keeps track of the seconds for you, so there is no reason to be concerned about them.

If you don't enter the time correctly MDOS will display an error message "Invalid Time" and wait for you to try again. If you make a mistake, or enter the wrong time, you will have the opportunity to change it again later.

The command: TIME can be used any time to recall or set the time.

Note: If you cannot save a file, try doing two back-to-back checks of TIME commands. If the results are grossly different, your clock could be in the TEST MODE. Entering a new time will reset the clock out of the test mode.

SETTING THE DATE

After entering the time you will see:

```
Current date is Tuesday 12-01-98
Enter new date (mm-dd-yy):_
```

If you don't enter a date that MDOS recognizes it will display an error message "Invalid Date" and wait for you to try again. If you make a mistake, or enter the wrong date, you will have the opportunity to change it again later.

To enter the date type in the numbers that represent the month, day, and year separated by hyphens, and then press the enter key. For example, to set the date to December 15, 1998 you would type and enter the following:

```
Current date is Tuesday 12-01-98
Enter new date (mm-dd-yy): 12-15-98
```

Instead of using hyphens you may also use a slash (/) to separate the numbers. Whichever you choose, if you don't enter the date the way MDOS recognizes it, MDOS will display an error message "Invalid date" and wait for you to try again. If you make a mistake, or enter the wrong date, you will have a chance to change it later.

After the current time and date have been entered correctly they may not have to be changed for an extended period of time. Inside the computer there is a battery that keeps the system clock powered even when the computer is turned off. When the system is turned on, simply pressing Enter will accept the current time and date stored in the system clock.

The command: DATE can be used at any time to recall or set the date.

START-UP MESSAGE

After you have entered the time and date, MDOS displays a start up message to identify itself and the version you are using, and waits for further instructions. The exact wording of the start-up message might be different from the following example depending on the version of MDOS you are using.

```
MDOS, Final Version 2.00
Copyright: 9640 News Contributors
Updated: March 07, 1994
15:44:21 Tuesday 11-01-94

A>_
```

THE SYSTEM PROMPT

The A> is called the system prompt, because the system program (MDOS) is prompting you to type a command. At this point MDOS is at command level; it is waiting for you to enter a command.

The system prompt also identifies the current drive, i.e., the drive where MDOS looks for a file. MDOS identifies drives by letter. Letters are assigned by logical drive order; the first drive in the system is drive A, the second is drive B, and so on. Four floppy disk drives will be referred to as A through D, and two fixed disks can be referenced as E and F.

If you're not using a fixed disk, your system prompt is programmed so that MDOS is loaded from drive A. MDOS thus assumes that drive A is the current drive, so the initial system prompt is A>. If you’re using a fixed disk, MDOS is loaded from the fixed disk. MDOS usually assumes drive E is the current drive, and the initial system prompt is E>.

COPYING THE MDOS DISKETTES

It is very important to copy your original diskettes to protect yourself from loss in case they are damaged. After copying (also called "backing up") your disks, you should also work from the copies, not the originals, for your protection.

If you have not made copies of your MDOS diskettes, follow the step-by-step procedures that follow. It is important that you copy your MDOS diskettes as soon as possible so that you don't risk damaging your originals. Remember, without MDOS you will be unable to operate your computer.

TWO-DRIVE BACKUP PROCEDURE

The following instructions assume that your MDOS system diskette is in drive A, that you have performed the start-up steps of entering the date and time, and that you have the MDOS system prompt, A>, on the screen.

1. Put a blank diskette in drive B.

2. Type the following command at the A> prompt and press Enter:

```
A>DISKCOPY a: b:
```

This command tells MDOS to copy everything on the diskette in drive A to the diskette in drive B. MDOS will respond:

```
Insert SOURCE diskette in drive A:

Insert TARGET diskette in drive B:

Press any key when ready...
```

3. If your source (MDOS) and target (blank) diskettes are in the appropriate drives, press any key to begin. MDOS will respond:

```
Copying 40 tracks
9 Sectors/Track, 2 side(s)

Formatting while copying
```

After the diskette has been copied MDOS will ask you if you would like to copy another diskette.

4. Remove the diskette from drive B and label it MYARC DOS.

5. Remove the original MDOS system diskette from drive A and put in a safe place. From this point on you will use your backup MDOS system diskette to avoid damaging the original one.

6. For each additional diskette that was included with your computer, respond Y to the following prompt:

```
Copy another diskette (Y/N)? _
```

7. Follow steps 1 through 5 for each additional diskette. These additional source diskette(s) will have different names and consist of different data, the new blank diskettes will also.

SINGLE-DRIVE BACKUP PROCEDURE

MDOS will prompt you to exchange the source and target disk in drive A several times. Follow the prompts, and remember that the MDOS diskette is the source and the blank is the target.

1. Type the following command at the A> prompt and press Enter:

```
A>DISKCOPY
```

MDOS will respond:

```
Insert SOURCE diskette in drive A:

Press any key when ready...
```

2. Your source (MDOS) diskette is already in drive A, so press any key to begin. MDOS will respond:

```
Insert target diskette in drive A:

Press any key when ready...
```

3. Remove the MDOS diskette, put in the blank diskette, and press any key to begin. MDOS will respond:

```
Formatting while copying
```

MDOS writes the data it read from the system diskette onto the blank diskette, and then it requests that you put the source diskette back in the drive:

```
Insert SOURCE diskette in drive A:

Press any key when ready...
```

4. Continue to exchange diskettes as MDOS prompts you to do so. After the last exchange, MDOS will respond:

```
Copy another diskette (Y/N)? _
```

5. Remove the new MDOS system diskette from drive B and label it MYARC DOS.

6. Put the original MDOS system diskette in a safe place. From this point on you will use your backup MDOS system diskette to avoid damaging the original one.

7. For each additional diskette that was included with your computer, respond Y to the "Copy another diskette" prompt.

8. Follow steps 1 through 7 for each additional diskette. These additional source diskette(s) will have different names and consist of different data, the new blank diskettes will also.

CHANGING THE CURRENT DRIVE

If you don't want MDOS to assume your files are on the diskette in drive A, you can change the current drive by typing the letter of the new drive followed by a colon. For example, to change the current drive to drive B, you would type and enter:

```
A>b:

B>_
```

The new system prompt will be B>, and MDOS will now assume your files are on the diskette in drive B unless told otherwise. If you are using a fixed disk and wish to change it to the current drive, you would type and enter:

```
B>e:

E>_
```

CHANGING THE TIME AND DATE

The computer has an internal clock that lets MDOS keep track of both the time of day and the date. If you're using the system when midnight arrives the date advances to the next day, and, if appropriate, the next month and year.

When the computer is off, a battery keeps the clock running so the time of the day and date are not lost. If you wish, you may alter the time and date whenever you start the system.

After MDOS has been started, you may check or change the date using the Date command. To tell MDOS you want to check or change the date, you would type and enter:

```
A>DATE
```

Just as when you start the system, MDOS displays the current date and prompts you for a new date. If you wish to leave the date unchanged press the Enter key. This allows you to check the current system date.

You can check and change the time the same way with the Time command. Type and enter:

```
A>TIME
```

CLEARING THE SCREEN

At times, when the screen is filled with commands and responses, you might want to clear it before continuing your work. You can erase everything on your screen with the Clear Screen (CLS) command. Type and enter:

```
A>CLS
```

After entering this command, the screen is cleared, except for the system prompt in the upper left corner.

PRINTING WHAT IS ON THE SCREEN

The screen display will show you a record of your commands and the responses from MDOS. When every line on the screen is filled, each additional line causes the entire screen to shift up, or scroll, to make room for the new line at the bottom.

A copy of what is on the display is often useful, especially when you are first learning MDOS. To have a copy of what is on the display sent to your printer, Print Screen key. Make sure your printer is turned on; a device error will occur if it is not. Also make sure PRN is set for your printer.

CONTROL KEY FUNCTIONS

Additional Control key combinations are used fairly often with MDOS. These key combinations are helpful in pausing the system, canceling a command, printing the same information that is being displayed on the screen, and restarting the system.

MDOS may not acknowledge all Control key commands on the screen, but when it does, it uses the caret symbol (^) in combination with a letter. Control-Break, for example, is shown on the screen as ^C, and can be typed by holding down the Control key and typing the letter C.

PAUSING THE SYSTEM

MDOS lets you temporarily halt system operation, and thus the display, by pressing Ctrl-Num Lock. Normally, this key combination is used to freeze the screen so you can read it or to suspend an invoked operation that you want to think about before allowing it to continue. Simply press any key to resume normal operations. Pressing Ctrl-S has the same effect as pressing Ctrl-Num Lock.

CANCELING A COMMAND

If you enter a command and then change your mind, or realize you meant to enter some other command, you can cancel the command by pressing Ctrl-Break. When you press Ctrl-Break MDOS

will stop what it is doing, display a ^C at the stopping point, and return to command level. Pressing Ctrl-C has the same effect as pressing Ctrl-Break.

PRINTING AND DISPLAYING SIMULTANEOUSLY

Pressing Print Screen will start printing whatever appears on the display. If you type, or if the system displays anything on the screen, it will be sent to the printer. MDOS will continue to print and display simultaneously until you press Print Screen again. Pressing Ctrl-P has the same effect as pressing Print Screen.

The Print Screen key can be a very handy device to produce a printed, or hard copy, historical log of operational procedure, error messages, and the like for future reference. Unfortunately MDOS waits until a line is printed before displaying and printing the next line, which causes the display to slow down.

RESTARTING THE SYSTEM

If you find yourself in a situation where you would like to start your system over from the beginning, without turning off the power, pressing Ctrl-Alt-Del will do so.

When this sequence of keys is pressed the screen will clear, the red light on drive A (or drive E if you have a fixed disk) goes on, and MDOS is loaded just as it was when you turned the power on. Restarting with the Ctrl-Alt-Del sequence takes less time then starting the system from scratch. This process is usually referred to as a "warm boot".

MDOS FUNCTION KEYS

While in MDOS, four of the cursor control keys have been designated to perform two very useful operations. The first two are used to scroll through the command stack one line at a time, and the last two will allow the screen to be scrolled.

SCROLLING THE COMMAND STACK

The command stack is a listing of the last few commands executed, usually consisting of no more than ten commands. After executing a command you may want to use it again. Instead of typing in the command from scratch, you simply press the up arrow to recall it.

Let’s assume you had just executed the following command and have been returned to the command prompt:

```
A>DISKCOPY a: b:

A>_
```

Pressing the up arrow would recall the last command as shown, and if the Enter key was pressed it would be executed again. If the up arrow was pressed a second time, the command executed before the last command shown would appear at the command prompt.

Pressing the down arrow would scroll forward through the command stack again, allowing you to easily move back and forth through previously used commands.

SCROLLING THE SCREEN

When the screen display scrolls upward it eventually causes the commands you have entered, and the responses from MDOS, to scroll off the display. The last screen, or page, that is scrolled off the display can be recalled using the Page Up key.

While in MDOS, pressing the Page Up key will cause the screen to scroll in reverse, line by line, and recall the last page of text that scrolled off the display. This will allow you to view any information that might have passed by too quickly before you had a chance to pause the display. To scroll the screen forward again, press the Page Down key. Arrow keys (Up & Down) also scroll one line at a time after using Page Up. If any other key is pressed you will be returned to the command prompt and the screen will be refreshed.

Pressing "P" while in the scrolling mode will print the screen.

TURNING THE SYSTEM OFF

If you are using MDOS, not an application program, all you have to do to shut the system down is turn off the power (in the proper sequence). This may be done at any time unless the red light on a disk drive is on. Turning the power off while a disk drive is in use can cause you to lose data on the disk.

As a precaution you may want to remove any diskettes you are using from the disk drives before turning off the system.

MANAGING YOUR FILES AND DISKS

The computer memory is temporary; it is cleared each time you turn off the computer. The only way you can save data permanently is to store the data in a file on a disk. When MDOS needs data that is stored in a file it reads the data from the disk into memory. Without disks and files, it would be very difficult and time consuming to use your computer.

TYPES OF FILES

In general, a file consists of either a program or data. A program is a set of instructions for the computer. Data is the text and numbers a program uses during its work, such as a table of tax rates, or a business letter.

Every file can be categorized as either a text file, command file, or an application program file. They are all different, so it's important to look more closely at the kind of information these files contain.

TEXT FILES

Text files are data files that contain characters you can read (everyday letters, numbers, and symbols.) Word-processing programs store their documents in text files, usually in Display Variable 80 format. Many files you use in your work with the computer are text files.

BATCH FILES

Batch files are special text files. They contain a series of MDOS instructions and/or program names. This makes it easy to perform repetitive tasks.

APPLICATION PROGRAM FILES

An application program, such as a word processor, is stored in a command file, or series of command files. Application files come in different forms depending upon the environment the application must be run under.

FILENAMES

Each file on a disk must have a different filename, regardless of its file type. A filename can be up to ten characters long, made up of any letter or number; you can also use the following symbols:

! # $ % ^ * ( ) - _ ` '

Try to make filenames as descriptive as possible. A short filename might be easy to type, but you may find it difficult to remember what the file contains if you haven't used it for a while. The more descriptive the name, the more easily you can identify the contents of a file.

SPECIFYING THE DRIVE

When you enter a filename, MDOS must know which drive contains the disk with the file on it. If you don't specify the drive letter, MDOS looks on the disk in the current drive (the drive letter shown in the system prompt). If the disk containing the file is not in the current drive, you can precede the filename with the drive letter followed by a colon. For example, if you specify the file as `b:manager`, MDOS looks for it in drive B.

WILDCARD CHARACTERS

To make it easier to manage your disk files, most commands let you use wildcard characters to handle several files at once. That way when you want to do the same thing to several files, you don't have to enter a separate command for each file. You can use the wildcard characters to tell MDOS you mean a set of files with similar names. Just as a wild card in a poker game can represent any other card in the deck, a wildcard character can represent any other character in a filename. Another term often used to reference wildcard characters is "global filename" characters.

THE ASTERISK (*) CHARACTER

The asterisk makes it easy to carry out commands on sets of files with similar names; it can represent all ten characters in a filename.

The following examples illustrate different ways to use the asterisk with selected commands. These commands will be fully explained later.

To get a listing of the directory entries where all filenames that begin with the letter M, type and enter the following:

```
A>DIR m*
```

To erase every file on the disk, type and enter the following:

```
A>ERASE *
```

THE QUESTION MARK (?) CHARACTER

The question mark replaces only one character in a filename. The asterisk will probably be used more frequently but the question mark can prove to be very useful.

To get a listing of the directory entries where the filenames last seven letters are MANAGER, regardless of the first three, type and enter the following: `A>DIR ???manager`

DISPLAYING THE DIRECTORY OF A DISK

The DIRectory command displays entries from the directory that MDOS keeps on each disk. Each entry includes the name of each file, its file type, its size in bytes, and the date and time it was created or last updated. You can use the DIRectory command to display all entries, or just the entries of selected files.

Four additional pieces of information are displayed when you ask for the directory of a disk. The two lines at the top specify the volume label, or disk name, and the drive and directory. The last line in the directory specifies the number of files on the disk that meet the criteria of the directory command entered and the amount of disk space left in bytes.

The format for the DIRectory command is as follows:

```
DIR [d:][filename][/W][/P]
```

- Include a filename and MDOS searches the disk in the current drive and displays the entry for that file.

  ```
  A>DIR manager
  ```

- Include both a drive letter and a filename and MDOS displays the entry for the file you specify from the disk in the drive you specify.

  ```
  A>DIR b:manager
  ```

- Include a filename with wildcard characters and MDOS displays the entries for all the files whose names match the wild-card characters.

  ```
  A>DIR m*
  ```

- If you omit a filename, but include a drive letter, MDOS displays all directory entries on the disk in the current drive.

  ```
  A>DIR b:
  ```

Because a list of directory entries can be quite long, the DIRectory command includes two options you can use to keep the list from scrolling off the top of the screen.

- /W (Wide) tells MDOS to display only the filenames in five columns across the screen. This display contains less information, because it omits the file types, sizes, dates and times, but makes a long list of entries more compact.

  ```
  A>DIR b: /W
  ```

- /P (Pause) tells MDOS to display the entries one full screen at a time. A message at the bottom of the screen will tell you to strike a key to continue.

  ```
  A>DIR b: /P
  ```

Appendix B shows additional parameters that may be used with DIR.

PREPARING A DISKETTE FOR USE

Before MDOS can store a file on a new diskette, you must prepare the diskette for use. The FORMAT command will be used to do this. Formatting a diskette erases any files that may have been previously stored on it, so be sure not to format a diskette that contains files you need.

In carrying out the FORMAT command, MDOS also checks for flaws on the recording surface of the diskette and marks any bad sectors so they won't be used. After formatting, MDOS will display a message that tells you the maximum number of sectors the diskette can hold, how many sectors (if any) are defective, and how many sectors are available for storing files.

MDOS knows whether drives are single-sided or double-sided and formats the diskette accordingly. If you have double-sided drives, but want to copy some files to a diskette for a friend who has single-sided drives, you can tell MDOS to format only one side of the diskette.

If you are using 80-track disk drives, MDOS will read and write from and to them without difficulty but the diskettes you write to can only be used on 80-track disk drives. The track formats used on 80-track disk drives are too narrow to be read reliably by standard 40-track disk drives.

MDOS automatically formats a diskette for nine sectors per track so it can be used with any system, regardless of the disk drive controller it is using. Diskettes can be formatted in both sixteen and eighteen sectors per track formats for those systems using double-density disk drive controllers. The TI disk controller can only format in single density, i.e., nine sectors per track. The MYARC disk controllers can format nine, and in double-density, both the recommended sixteen, ***and*** eighteen sectors per track. The CorComp disk controller will only format nine and eighteen sectors per track.

The FORMAT command reserves space on the diskette for the directory, thus reducing the amount of storage available for files. There is no reason to be alarmed when you notice that you may not directly utilize the entire disk for storing your files.

The parameters for this command are as follows:

```
FORMAT [d:] [/1] [/16][/18] [/36][/80] [/N][/V] /Sides /SectorsPer
Track /Tracks /N /V     (Default is /2/9/40)
```

- Omit the drive letter and MDOS will attempt to format the diskette in the current drive.

  ```
  A>FORMAT
  ```

- If you include a drive letter, MDOS will attempt to format the diskette located in the drive specified.

  ```
  A>FORMAT b:
  ```

- Specify /V and MDOS will prompt you to enter a volume label, or disk name, for the disk that is going to be formatted.

  ```
  A>FORMAT b: /V
  ```

- Specify /1 and MDOS will format the target diskette for single-sided use, regardless of the drive type.

  ```
  A>FORMAT /1
  ```

- Specify /16 or /18 and MDOS will format the target diskette in either the sixteen or eighteen sectors per track format.

  ```
  A>FORMAT /16
  ```

Example: Formatting a Diskette

The following is an example of formatting a diskette in drive B, and giving it a volume label.

```
A>format b: /v
```

MDOS asks you to put the diskette in drive B:

```
Insert new diskette in drive B: and strike ENTER when ready_
```

MDOS then prompts you for a volume label:

```
Volume label (10 characters, ENTER for none)? _
```

You may type and enter any volume label you desire. If you just press enter the diskette will have a volume name of "none". After MDOS has completed the formatting process, it will display a report of available storage on the diskette, and ask you if you want to format another diskette.

The /N switch disables floppy verification of all sectors and speeds up formatting.

See Appendix C for information on formatting 1.44mb drives and RAMdisks.

COPYING A COMPLETE DISKETTE

Although an entire diskette can be copied using a combination of the Format and Copy commands, the DISKCOPY command simplifies this process. DISKCOPY can be used to copy the entire contents of one diskette (the source diskette), to a new diskette (the target diskette). It will also format the target diskette according to the format used on the source diskette. The DISKCOPY command only works with diskettes; you cannot use it to copy to or from a hard disk.

The format is as follows:

```
DISKCOPY [d:] [d:]
```

The first parameter specified is the source drive. The second parameter is the target drive.

- Omit the target drive and MDOS copies from the diskette in the source drive to the diskette in the current drive.

  ```
  B>DISKCOPY A:
  ```

- If you omit the target drive and specify the current drive as the source drive, MDOS assumes you want to use only the current drive (single drive copy) and prompts you to switch diskettes during the copy process.

  ```
  A>DISKCOPY A:
  ```

- If you don't specify a source or target drive, MDOS assumes you want to use only the current drive (single drive copy) and prompts you to switch diskettes during the copy process.

  ```
  A>DISKCOPY
  ```

The DISKCOPY command was used when copying your original MDOS diskettes in the previous section, "Starting MDOS".

COMPARING TWO DISKETTES

You may want to know if two disks are identical. The DISCOMP command compares two diskettes sector-by-sector to determine if any differences exist. The DISKCOMP command may only be used with diskettes. You cannot use it to compare a fixed disk to a diskette.

The format for the DISKCOMP command is as follows:

```
DISKCOMP [d:] [d:]
```

- If you omit the second drive letter, MDOS will compare the diskette in the first drive specified to the diskette in the current disk drive.

```
B>DISKCOMP a:
```

- If you omit both the first and second drive letters, MDOS assumes you want to use only the current drive and prompts you to switch diskettes during the comparison.

```
A>DISKCOMP
```

If MDOS finds any differences between the two diskettes being compared, it displays the side and track of each; for example:

```
Compare error on side 0, track 27
```

Example: Comparing Two Diskettes

To compare the diskette in drive B to the diskette in drive A, you would type and enter:

```
A>DISKCOMP a: b:
```

MDOS will prompt you to insert the diskettes:

```
Insert FIRST diskette in drive A:
Insert SECOND diskette in drive B:

Press any key when ready...
```

After you begin the comparison, MDOS will report how many tracks, sectors, and sides it is comparing. It will then report the results of the comparison and asks if you want to compare more diskettes.

```
Comparing 40 tracks
9 sectors per track, 2 side(s)

Compare: OK

Compare another diskette (Y/N)? _
```

DISPLAYING A FILE

Many files that you use, such as word processing files, are text files. There will be times when you want to check the contents of a file without having to load a special program to read or print the file. MDOS gives you a quick way to see what's in a file; the TYPE command.

When you use the TYPE command, MDOS displays the file without stopping. If you would like to read the file but its longer than one screen, freeze the display using Ctrl-Num Lock or Ctrl-S or use the /M switch and each screen will stop with "Press any key, (A)bort, or (N)onstop...".

The format for Type is as follows:

```
TYPE [d:][filename][/M]
```

The filename is the name of the file to be displayed. The TYPE command can display only one file at a time, so you can't use wildcard characters in the filename. If you do use a wildcard character, or if the file you choose doesn't exist, MDOS displays the message "Invalid filename or File not found" and returns to command level.

For example, to display the file named LETTER on the diskette in the current drive you would use the following command:

```
A>TYPE letter
```

MAKING COPIES OF FILES

Just as you sometimes make copies of your paper files, you'll find yourself needing copies of disk files. The COPY command can make a copy of a file on the same disk (with a different filename) or on a different disk (with the same filename, if you wish).

When used to make copies of files, the COPY command has two major parameters, the original file and the new file to be created.

To copy files, use the following command:

```
COPY [d:][filename] [d:][filename]
```

The first file specification is the name and location of the file to be copied (the source file). The second file specification is the name of the copy to be made and its location (the target file). Wildcard characters can be used to copy a set of files more easily.

- If you specify a drive other than the current drive as part of source file and omit the target file, the file is copied to the disk in the current drive and is given the same name as the source file.

  ```
  A>COPY b:manager
  ```

- If you specify only a drive letter as the target file, the file is copied to the disk in the drive you specify and is given the same name as the source file.

  ```
  A>COPY a:manager b:
  ```

- If you specify the source file that doesn't exist, MDOS responds:

  ```
  "<filename> File not found
  0 File(s) copied"
  ```

  and returns you to command level.

- If you specify the source file that is not on the disk in the current drive and omit the target file, MDOS copies the source file to the disk in the current drive and gives the copy the same name as the original.

  ```
  A>COPY b:report
  ```

- If the target file that doesn't exist, MDOS creates it.

- If you specify a target file that does exist, MDOS replaces it with the source file. This is the same as erasing the existing file, so be careful not to give a copy the same name as an existing file you want to keep.

Example: Copying Files

The following is an example of copying the file MANAGER from drive A to drive B.

```
A>COPY a:manager b:manager
```

After the copy has been made, MDOS will acknowledge a successful copy by displaying:

```
"1 File(s) copied".
```

Filenames with a slash (/) in them must be enclosed in quotes.

Note: A bug in MDOS does not check for space on the target disk when copying D/V80 files--be especially careful using the wildcard *.

ERASING FILES

Just as you have to clean out a file drawer once in a while, you'll occasionally have to clear your disks of the files you no longer need. The ERASE command erases (deletes) files from a disk and is interchangeable with the DELete command. Using the wildcard characters, you can erase a particular set of files with a single command.

To delete files:

```
ERASE [d:][filename]
```

or

```
DEL [d:][filename]
```

When erasing files, if you:

- Omit the drive letter and MDOS will erase the file(s) specified on the disk in the current drive.

```
A>ERASE report
```

- If you use wildcard characters, MDOS will erase all files that match the filename and wildcard specifications.

  ```
  A>ERASE b:m*
  ```

- If you specify a file that does not exist, MDOS responds "File not found", and returns to command level.

Whenever you use the ERASE command with wildcard characters, you should double check the command on the screen before you press the ENTER key. It is very easy to erase files you do not want to remove from a disk when using wildcard characters. Make sure you have specified the correct drive letter (if necessary) and filename. Know exactly which files you are going to erase.

If you tell MDOS to erase all the files on a disk by typing and entering:

```
A>ERASE *
```

MDOS will prompt you, "Are you sure (Y/N)?" before erasing the files. If you respond with anything other than "y", MDOS then cancels the ERASE command and returns to command level.

CHANGING FILENAMES

There are times when you'll want to change the name of a file. You may simply change your mind, or perhaps have changed the contents of a file so much that you want to give it a new name that more closely describes its new contents. The REName command changes a file's name. Wildcard characters can be used to rename a set of files.

The Rename command's format is as follows:

```
RENAME [d:][filename] [filename]
```

or

```
REN [d:][filename] [filename]
```

The first filename is the name of an existing file to be renamed. If the file doesn't exist, MDOS responds with the message "Duplicate filename or File not found" and returns to the command prompt.

The second filename is the name you want to give to the file specified by the first filename. For example, to change the file OLDNAME on drive A to NEWNAME you would type and enter the following:

```
B>RENAME a:oldname newname
```

If there is already a file with the new name, MDOS displays the message "Duplicate filename or File not found" and returns to command level. Two files on the same disk can't have the same name.

CONTROLLING WHETHER A FILE CAN BE CHANGED

Your disks will contain many files. Some, such as program files, you will seldom, if ever, erase. Although you probably have backup copies, some of these files may only exist on your working disks; erasing them could represent a serious loss.

The ATTRIBute command lets you protect yourself from inadvertently erasing or changing a file by making it read-only. Before a read-only file can be changed or erased, the protection must be removed with the ATTRIB command.

The ATTRIB command takes the following parameters

```
ATTRIB [+/-P][d:][filename]
```

- If you use +P before the file specification, MDOS will deny all attempts to change or erase the filename in the future.

  ```
  A>ATTRIB +P manage
  ```

- If you use -P before the file specification, MDOS will let the file be changed or erased in the future.

  A>ATTRIB -P manage

- If you enter the command with just the filename, MDOS displays the name of the file, and if the file is read-only, a P appears in the first column.

  ```
  A>ATTRIB report
  ```

  MDOS responds:

  ```
  P DSK1.report
  ```

CHECKING THE CONDITION OF A DISK

The Check Disk command (CHKDSK) allows you to analyze the directory on a disk, comparing the directory entries with the locations and the lengths of the files, and reports any errors it finds. The CHecK DiSK report includes the following:

- The total amount of space on the disk.
- The number of files and directories, and how much space they occupy.
- How much space on the disk remains available for files.
- The size of the computer memory and how many bytes remain free for use.

You can also have the command display the name of each file on the disk to check whether any files are stored inefficiently.

If possible, MDOS stores files in adjacent, or contiguous, sectors. As files are deleted and new files are stored they can become fragmented. A fragmented file isn't a cause for worry; the worst that can happen is that MDOS will take slightly longer to read the file. If several files on a diskette are fragmented, you can restore them to contiguous sectors by copying all the files to an empty, freshly formatted diskette with the Copy command.

The CHKDSK command takes the following parameters:

```
CHKDSK [d:][filename][/F]
```

- If you omit the drive letter, MDOS checks the disk in the current drive.

  ```
  A>CHKDSK
  ```

- If you include a drive letter, MDOS checks the disk in the drive specified.

  ```
  A>CHKDSK b:
  ```

- If you include the name of a file whose storage you want to check, MDOS will display a message if the file is stored in non-contiguous sectors. Wildcard characters can be used to check a set of files.

  ```
  A>CHKDSK report
  ```

- If you specify /F, MDOS will automatically correct any errors it finds in the directory when the error is found.

  ```
  A>CHKDSK /F
  ```

If the CHKDSK command finds an error, it displays a message, such as "Disk error reading drive B" or "Allocation error, file size adjusted", followed by a filename and a prompt asking you whether to correct the error. If you specified the /F parameter, you can reply "y" to tell MDOS to try to correct the error. Depending on the type of error, this may cause the loss of some data.

Examples: Checking a Disk

To check a diskette in drive B, type and enter the following:

```
A>CHKDSK b:
```

MDOS will display its report:

```
327680 bytes total disk space
512 bytes in 1 directories
```

```
262144 bytes in 36 user files
65024 bytes available on disk

524288 bytes total memory
487248 byes free
```

To check a diskette in drive B, and to check whether all the files on it are stored in contiguous sectors, type and enter:

A>CHKDSK b:*

MDOS will display a report like the one above, but adds the following message:

```
All specified files(s) are contiguous
```

If any files were stored in non-contiguous sectors, MDOS would display their names in place of the above message.

ASSIGNING OR CHANGING A DISK'S VOLUME LABEL

The LABEL command assigns, changes, or deletes the 10 character volume label of a diskette or of a fixed disk. Its format is as follows:

```
LABEL [d:][volume label]
```

- If you omit the drive letter, MDOS assumes you want to alter the volume label of the disk in the current drive.

  ```
  A>LABEL
  ```

- If you omit the 10 character volume label, MDOS prompts you to enter a new label.

  ```
  A>LABEL b:

  Volume label (10 characters, ENTER for none)?
  ```

DISPLAYING A DISK'S VOLUME LABEL

The VOLume command displays the volume label of a fixed disk or diskette. If you assign descriptive volume labels to your diskettes when you format them, you can use the volume command to make sure you are using the correct diskettes. It's faster and easier then checking the directory.
To use the VOLume command:

```
VOL [d:]
```

The drive letter designates the drive that contains the diskette whose volume label is to be displayed. If the drive letter is omitted, MDOS displays the volume label of the disk in the current drive.

For example, to display the volume label of a diskette in drive B, type and enter:

```
A>VOL b:
```

FIXED DISK ORGANIZATION

Until now we have avoided mentioning one of the key features of MDOS; its ability to have a hierarchical directory structure. For a floppy disk based system, you will normally use a single directory because of diskette storage capacity limitations.

To make your computer filing system more flexible, MDOS lets you create additional directories, called subdirectories, on a disk. The subdirectories divide the disk into different storage areas, each of which can be used as if it were a different disk.

To distinguish the main directory that MDOS creates from the subdirectories that you create, the main directory is known as the root directory because, as you will see, a multilevel directory structure can grow from it.

As you add levels to your file structure, a block diagram would show it spreading from the root directory and branching to other directories, like a tree branches from its root. This type of file structure is often called a tree-structured file system.

SUBDIRECTORIES

A subdirectory is a simple file that contains directory entries; these entries are identical in form to the entries in the main directory. You name a subdirectory as you would a file, but because a subdirectory defines other files, you cannot use the normal commands to copy or erase a subdirectory.

DIRECTORY AND PATHNAMES

Subdirectory names follow the same conventions as standard MDOS filenames; that is, they can be up to ten characters in length. It is a good idea to give the subdirectories a name that best reflects its contents.

Pathnames specify the route through a directory structure required to locate a specific directory or file. Pathnames start at the root directory, and branch out through the subdirectory tree structure. The first backslash (\) character refers to the root directory.

In the following tree-structured directory example (Figure 6), the root directory is located at the top of the tree. Each subdirectory is given a name, and the last four subdirectories contain two files each.

Keep in mind that this is only a simple example. A fixed disk can contain a large array of subdirectories, each containing files of its own and additional subdirectories. We will use this simple example to help you illustrate how to make, create, change, remove, and utilize subdirectories.

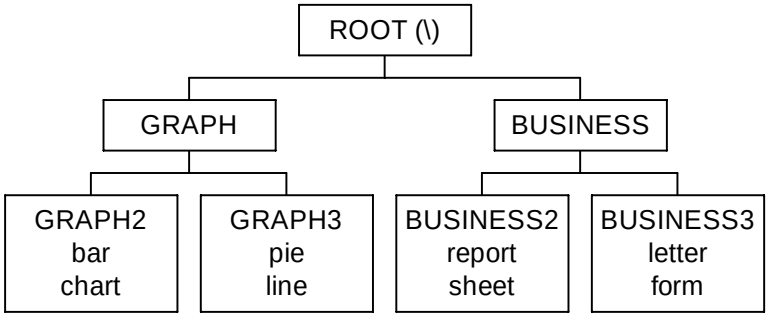

Figure 6. Tree Structured Directory consisting of a root directory, subdirectories (upper-case), and files (lower-case).

CREATING SUBDIRECTORIES

The Make Directory (MKDIR, or MD) command creates a subdirectory. The Make Directory command's format is as follows:

```
MKDIR [d:]path (where path is the new subdirectory-9 chars max)
```

or

```
MD [d:]path
```

- If you do not specify a drive, MDOS creates the subdirectory on the disk of the current drive.

  ```
  E>MD graph
  ```

- Enter a path of the directory in which the subdirectory is to be created and the subdirectory will be made in the directory specified by the path.

  ```
  E>MD graph\graph2
  ```

- If you omit the path, the subdirectory is created in the current directory. If the current directory was the root (\) directory it would be created there.

  ```
  E>MD business
  ```

If you had actually created these subdirectories you could see them by displaying the root directory. You could display the root directory using the DIRectory command. It would display the two directories just created, and any files that may be present in the root directory.

```
E>DIR
```

MDOS would display:

```
Volume in drive E is OFFICE
Directory of E:\

GRAPH   <DIR>  12-15-86 11:15a
BUSINESS <DIR>  12-15-86 11:15a
2 file(s) 9355488 bytes free
```

Note that the directory identifies the files as subdirectories by displaying <DIR> after their names. The backslash (\) in the second line of the display is what MDOS uses to refer to the root directory of a disk.

Not shown on the display are the GRAPH2, GRAPH3, BUSINES2, and BUSINES3 subdirectories. Those directories were placed, using a pathname, as subdirectories in the GRAPH and BUSINESS directories.

THE PATH TO A DIRECTORY

Almost every command can include a pathname. For example, to display the GRAPH directory from the root directory you could use the following command:

```
E>DIR \GRAPH  (for floppies prob. need B>DIR\GRAPH\ )
```

MDOS would display:

```
Volume in drive E is OFFICE
Directory of E:\

.       <DIR>  12-15-86 11:15a
..      <DIR>  12-15-86 11:15a
GRAPH2  <DIR>  12-15-86 11:17a
GRAPH3  <DIR>  12-15-86 11:17a
2 file(s) 9355488 bytes free
```

Two mysterious directories seem to appear in the GRAPH directory, and will appear in every subdirectory you create. These dot directories (. and ..) are directory markers which are designed to let you move quickly up and down a directory structure. The double periods represent the directory that contains the current directory.

A pathname can also be included with a file name, to tell MDOS where to find a file. The pathname goes just before the filename (after the drive letter, if one is included) and is separated from the filename by a backslash. For example, if the subdirectory \GRAPH2 contained the file named BAR, the path to the filename would be \GRAPH\GRAPH2\BAR.

Pathnames can prove to be very valuable in helping you manage your files more easily with commands such as COPY, ERASE, REName, and the like.

CHANGING THE CURRENT DIRECTORY

Just as MDOS keeps track of the current drive, it also keeps track of the current, or working, directory. When you start MDOS, the current drive is the drive from which MDOS was loaded; the current directory is the root directory of the current drive.

Just as you change the current drive, you can change the current directory, so that you don't have to type the pathname each time you want to work with a directory other than the current directory.

The CHange DIRectory (CHDIR, or CD) command displays the name of, or changes the current directory. The CD command's format is as follows:

```
CHDIR [[d:]path]  (to change to root use CHDIR[d:] \ )
```

or

```
CD [[d:]path]   (to change to root use CD[d:] \ )
```

- If you omit the drive letter, MDOS changes the current directory on the disk in the current drive.

  ```
  E>CD graph
  ```

  Note: An trailing \ is optional.

Examples: Changing the Current Directory

The following command would change the current directory of drive E from \ to GRAPH2:

```
E>CD graph\graph2
```

Using the subdirectory markers, the following command would change the current directory of drive E from GRAPH2 to GRAPH:

```
E>CD ..
```

REMOVING A SUBDIRECTORY

As you work with a multilevel directory structure, you may find that you no longer need a particular subdirectory. The Remove Directory (RMDIR, or RD) command removes a subdirectory. A subdirectory cannot be removed if it contains any files or subdirectories.

To use Remove Directory:

```
RMDIR [d:]path
```

or

```
RD [d:]path
```

- If you do not specify a drive, MDOS removes the subdirectory on the current directory on the disk of the current drive.

  ```
  E>RD graph
  ```

- If you enter a path, it will remove the last directory specified.

  ```
  E>RD graph\graph2
  ```

- If you omit the path, MDOS removes the subdirectory on the current directory on the disk of the current drive.

  ```
  E>RD business
  ```

If the pathname is incorrect, or if the directory does not exist, or if files or subdirectories remain in the directory that you wish to remove, MDOS will not remove the current directory. Instead, MDOS will respond with "Invalid Path", "Directory not found", or "Directory not empty"

USING SUBDIRECTORIES

Manipulating files and maneuvering through subdirectories is not that complicated. Just view each directory as if it were a separate disk.

The examples that follow will help give you a general understanding of how to use subdirectories. These are only examples. There are literally hundreds of command entries that are possible with respect to subdirectories.

To display the contents of the root directory from any (current) directory, you would use the following command:

```
E>DIR \
```

Likewise, you can display the contents of any directory, provided the proper pathname is specified. For example, if your current directory is BUSINESS2, you can display the contents of the GRAPH2 directory by using the following command:

```
E>DIR \GRAPH\GRAPH2
```

It's easy to copy a file from one directory to another. If the current directory is BUSINES2, you can copy the file BAR from the GRAPH2 directory to the BUSINES2 directory with a new filname of NEWBAR, using the following command:

```
E>COPY \GRAPH\GRAPH2\bar newbar
```

Just as MDOS doesn't confuse two files with the same name on different disks, it doesn't confuse two files with the same name in different directories. MDOS can tell the latter apart because their paths are dissimilar. You can demonstrate this by copying the file named REPORT from the current directory (BUSINESS2), to the subdirectory BUSINESS3. The following command can be used:

```
E>COPY report \BUSINESS\BUSINESS3
```

The target filename does not have to be included if you wish to give it the same name as the original file.

THE PATH TO A COMMAND

In a multilevel filing system, you'll probably change the current directory as you use different files in the different subdirectories. You will undoubtedly also use command files, such as external MDOS commands, that are outside of your current working directory. When you type a command, MDOS looks for the command file in the current directory, if the command you have entered does not appear in the current directory, MDOS will display the message "Bad command or Filename".

The PATH command lets you tell MDOS where to look for a command file if it's not in the current directory. You can name one or more directories on any disk drive. It is most common to name the root directory and a specific subdirectory of commonly used commands as your path.

The Path command has the following format:

```
PATH [[d:]path[[;[d:]path]...]]
```

Where path is a directory name or filename.

- If you enter the Path command with no parameter, MDOS will display the current pathnames that were specified in a previous path command.

  ```
  E>PATH
  ```

- If you enter the Path command with only a semicolon, MDOS resets the search to null. MDOS will only search the current directory for a command. This is the default set when MDOS is first started.

  ```
  E>PATH ; (note 1 space between H and ;)
  ```

- If you omit the drive letter, MDOS searches the directory specified by the path on the current drive.

  ```
  E>PATH graph
  ```

- If you omit the path, but include the drive letter, MDOS will search the current directory of the drive specified.

  ```
  E>PATH a:
  ```

You can specify several command paths in one command, separating them with semicolons. If you specify a large number of paths to be searched, MDOS will take a longer period of time to return you to the command prompt if the command or filename is not found.

Examples:

```
>PATH A:;B:;C:;  >PATH DSK1.;DSK2.;DSK3.;
```

DISPLAYING THE DIRECTORY STRUCTURE

If you create a file structure with several levels, you may not remember exactly what subdirectories you have created or exactly where they are. The TREE command displays the path of each subdirectory on a disk and displays the name of each file in each subdirectoy.

The Tree command uses the following parameters:

```
TREE [d:]
```

- If you omit the drive letter, MDOS will display the directory structure of the disk in the current drive.

  ```
  E>TREE
  ```

- Listing may be limited to showing only certain types of files by the use of the ` (Appendix B).

USING BATCH FILES

MDOS gives you a great deal of control over your computer system using a large assortment of commands. Because many people use MDOS for different purposes, and use different commands more than others, MDOS lets you create powerful commands tailored to your specific needs.

Creating a command of your own is simple. Using a text file (called a batch file), commands can be developed and saved using any name of your choice (except the name of an existing command.) The command can be used after it is saved by typing the name of the batch file at the command prompt; MDOS carries out the command(s) the file contains as if each had been typed in separately. Commands created in this manner are called batch commands.

HOW MDOS SEARCHES FOR A COMMAND

If you type something at the command prompt, MDOS assumes you have entered a command name. It then follows a particular sequence in trying to carry out the command:

1. It checks to see if you entered the name of a built-in command, such as DIR or COPY. If you did, MDOS executes the command.

2. If what you entered is not a command, MDOS checks to see if you entered the name of an application program file. If you did, MDOS would load and run the program file.

3. If what you entered is not the name of a program file, MDOS checks to see if you entered the name of a batch file (indicated as a Display/Variable 80 file type). If you did, MDOS carries out the commands in the batch file.

This sequence is important, because it explains why certain commands will be carried out before others. For example, if you named a batch file COPY, MDOS would never execute it because COPY is a command and would be executed first.

CREATING A BATCH FILE

A batch file can be created easily using the PE function of MY-WORD. Just as you would create and edit a normal text file, a batch file can be created and edited in the same manner.

When a file is created using the PE function of MY-WORD using a left margin of 0 and leaving no blank lines, it may be saved with SF in the Display/Variable 80 format. For example, to save the batch file COPYAB you would first enter SF to save a file, then enter A:COPYAB. The batch file COPYAB would then be saved in the correct batch format on drive A.

Example: Creating a Batch File

The following is a step-by-step example of creating a batch file for use within MDOS. This batch file would format a disk in drive B, and then proceed to copy all of the files on the disk in drive A to the newly formatted disk in drive B.

Enter the PE mode of MY-WORD and type in the following as it is shown below: (Remember 0 left margin and no skipped lines).

```
FORMAT b:
COPY a:* b:*
```

After you have entered the two commands, exit the Edit mode. Enter SF to save the file and enter the following:

```
DSK1.COPYAB
```

The file will be saved to drive A in batch format, without the tab line.

BATCH FILE COMMANDS

Virtually every MDOS command can be used as a command within a batch file. There are also additional commands, all are internal, that are usually used exclusively in batch files.

DISPLAYING MESSAGES FROM A BATCH FILE

The REMark command doesn't cause MDOS to do anything, but it is a valid command. However, the REMark command allows you to display a message from a batch file. Its format is as follows:

```
REM [message]
```

All remark messages are displayed when the execution of a batch file reaches the remark. Each remark can be up to 80 characters in length. You can also use the REMark command without any remarks for spacing within your batch file.

If the following Remark command was issued in a batch file, this remark would be displayed:

```
REM This batch file will copy a disk in drive A to B
```

CONTROLLING SYSTEM MESSAGES

You may occasionally create a lengthy batch file and use a liberal number of Remark commands as a mechanism to internally document the file. In such case, you may wish to turn off the display of Remark statements as well as other MDOS commands executed from a batch file. This can be accomplished by the use of the Echo command.

The ECHO command can be used to inhibit, or enable the display of MDOS commands executed from a batch file as well as display a message. Its format is as follows:

ECHO [on/off][message]

- If you turn ECHO on, MDOS will display commands as they are carried out. Echo is on unless you turn it off with the off parameter.

  ```
  ECHO on
  ```

- If you turn ECHO off, MDOS will not display commands as they are carried out. Eliminating the commands from the display can make a batch file easier to use by reducing the clutter on the screen.

  ```
  ECHO off
  ```

- If you include a message, ECHO will display a string of characters whether ECHO is turned on or off. The first character will be displayed at the left edge of the screen. If the messag starts with any spaces, they will be eliminated.

  ```
  ECHO Warning: All data will be erased on drive B
  ```

- If you omit all parameters the ECHO status (on or off) will be displayed.

MAKING THE SYSTEM PAUSE

Some MDOS commands, such as FORMAT and DISKCOPY, display a message and wait for you to respond, giving you a chance to confirm your intention or complete preparation by inserting a diskette or turning on the printer. You can have your batch files do the same; use the PAUSE command, which displays its own built-in message. "Strike any key when ready..." and makes the system wait until you press any key. Its format is as follows:

```
PAUSE [message]
```

An optional message can be included and will be displayed upon the execution of the PAUSE commands. The message will be displayed only if ECHO is turned on.

If the following PAUSE command was issued in a batch file, this message would be displayed:

```
PAUSE Change diskette in drive A

Strike any key when ready...
```

CONTROLLING WHICH COMMANDS ARE CARRIED OUT

Besides carrying out MDOS commands as though you had typed them in individually, batch files let you specify that a command should be carried out only if some condition is true. This capability makes your batch files more flexible, letting you adapt to a variety of situations.

The IF commands specifies the condition to be checked and the MDOS command to be carried out. Its format is as follows:

```
IF [NOT] condition command
```

When the IF parameter's condition is true, then the MDOS command is executed. When a NOT is included, the command is carried out only if the condition is not true. Otherwise the MDOS command is skipped, and the NEXT command in the batch file is executed.

The condition parameter will be checked before the command specified can be carried out. It has two commonly used forms:

- EXIST *filename* checks whether the named file exists. A path name can be specified, if necessary. If the filename exists, the condition is true.

  ```
  IF EXIST report GOTO LOOP
  ```

- String1==String2 compares two character strings. If they are identical, the condition is true. Note that there are two equal signs.

  ```
  IF %1==report ECHO report found!
  ```

The command used can be any valid MDOS command.

CHANGING THE SEQUENCE OF COMMANDS

Batch file commands ordinarily are carried out in the order in which they appear in the batch file. If you could control the order in which the commands are carried out, your batch commands would be more flexible. The GOTO command gives you this control by telling MDOS to go to a specific line in the command file, rather than to the next command in the sequence.

You tell MDOS where to go in the batch file by specifying a label. A label identifies a line in a batch file; it consists of a colon (:) immediately followed by a string of characters (such as :START). A label is

not a command – it merely identifies a location in a batch file. When MDOS transfers itself to a label, it carries out whatever commands follow the label.

```
GOTO label
```

The label identifies the line in the batch file to be transferred to. If the label is not found, MDOS terminates the batch file being executed with the message "Label not found". A label's first eight characters are significant, while the characters that follow are not. Labels within a batch file are never displayed while the batch file is executing.

In this example, the following batch file produces an indefinite sequence of "rem looping..." and GOTO LOOP messages on the screen:

```
:LOOP
rem looping...
GOTO LOOP
```

CARRYING OUT A COMMAND MORE THAN ONCE

At times you may want MDOS to carry out a command more than once in a batch file. The FOR command would be used to do so. Like the IF command, the FOR command has two parts: the first part defines how often the command is to be carried out, and the second part is the command to be carried out.

The format of the FOR command is as follows:

```
FOR %%variable IN (set) DO command
```

The %%variable is sequentially set to each member of (set) and then the command is evaluated. If a member of (set) is an expression involving * and/or ?, then the %%variable is set to each matching filename from disk.

It is less complicated then it sounds. When MDOS carries out a FOR command, it assigns to %%variable (for instance %%f), in turn, each value you specify in (set), then carries out the command. Each time it carries out the command, it first substitutes the current value taken from (set) for %%variable.

In this example, if you entered the command:

```
FOR %%f IN (one two three) DO ECHO %%f
```

Provided ECHO was off, the result would be:

```
one

two

three
```

BATCH FILE PARAMETERS

Many MDOS commands include one or more parameters to make them more specific. When you enter the DIRectory command, for example, you can specify the filename, to display some portion of the files on a disk, and the /W option, to display the wide form of the directory. Parameters let you use the same MDOS commands different ways. You can give your batch files the same flexibility by defining their own parameters.

A parameter is represented in a batch file by using a special symbol. When you use the batch file, MDOS replaces the symbol with a parameter you include when you enter the batch command. The symbol consists of a percent sign followed by a one digit number, such as %1. You can use the numbers 0 to 9 as parameters, and more than one parameter can be used within a single batch file.

The number of the symbol identifies which parameter replaces the symbol. If a batch file takes two parameters, for example, MDOS replaces %1 wherever it occurs in the batch file with the first parameter you enter, and it replaces %2 with the second parameter you enter.

Example: Using Batch File Parameters

The following batch file, named CF, was created to copy the filename on drive A specified by the first parameter to drive B when given the filename specified in the second parameter:

```
COPY A:%1 B:%2
```

This command could be entered to copy REPORT from drive A to drive B with the new file having the same name as the original file:

```
A>CF report *
```

CHAINING BATCH FILES

A batch command can be used in a batch file that defines another batch command. The second batch file could contain another batch command, and so on. This is called chaining, because each batch file is linked to the next one.

When a chained batch command is found in a batch file, MDOS jumps to the new batch file and will ignore all commands that might follow the batch command. Any commands that follow the chained batch command in a batch file would not be carried out since MDOS does not return to a previous batch file.

Chaining batch files is a good way to build powerful batch commands because it lets you use one batch file in several different ways and reduces the chance of your batch files getting long and complicated.

CANCELING A BATCH COMMAND

Just as with other MDOS commands, pressing Ctrl-Break or Ctrl-C will cancel the execution of a batch command.

When you cancel a batch command, MDOS will prompt you to confirm termination with the prompt "Terminate batch job (Y/N)?". If you respond no, the command being carried out is canceled, but MDOS continues with the next command in the batch file. If you respond yes, MDOS cancels the entire batch command and returns you to the system prompt.

THE AUTOEXEC BATCH FILE

Each time you start or restart the system, MDOS prompts you for the date and time; this is its start-up procedure. It is really just carrying out the Date and Time commands. You can substitute your own start-up procedure by creating a special batch file named AUTOEXEC that contains commands you always enter when you turn on your system.

Each time MDOS is started it goes through a short set of system diagnostic tests. After these tests are completed, MDOS checks to see if there is a batch file named AUTOEXEC on the system disk. If there is not, MDOS carries out the Date and Time commands and displays the system prompt. If there is an AUTOEXEC batch file on the system disk, MDOS carries out the commands in the AUTOEXEC batch file, and skips the Date and Time commands.

The AUTOEXEC batch file can be used in a variety of ways. For example, you could reset the system prompt, reconfigure the system, and load up a RAM disk automatically each time MDOS is loaded.

MANAGING YOUR DEVICES

Data flows into and out of a computer system through pieces of equipment called devices. Devices are categorized by whether they handle data coming in (input) or going out (output), or both. The keyboard, for example, is an input device; the computer gets information from it. A printer is an output device; the computer sends information to it. A disk drive is both an input device and an output device; the computer can either read a file from a disk or write a file onto a disk.

Some devices, such as the keyboard, don't need much attention from you, because MDOS requires no special instructions to operate them. Other devices, such as a color display or a printer, sometimes require you to tell MDOS how you want to use them. Color displays, printers, and the computers communications channels, called ports, can be used in a variety of ways.

DEVICE NAMES

Just as files have names, so do devices. You can use a device name in many MDOS commands just as you would use a filename. MDOS assigns all device names, thus you cannot name a device yourself.

CON is short for Console. It is both an input device and an output device, and refers to both the keyboard (input) and the display (output). Because the keyboard is an input only device, and the display is an output only device, MDOS can tell which one to use by the way you use the name CON in a command.

PRN is short for Printer. It is an output device, and refers to the parallel printer that MDOS uses unless you specify otherwise. You can attach as many as two parallel printers, named PIO/1 and PIO/2. MDOS assumes that PRN means PIO/1 (or just PIO) unless you or an application specify otherwise.

AUX is short for Auxiliary. It is for both input and output, and refers to the communications port that MDOS uses unless you specify otherwise. You can attach up to four communications ports, named RS232/1, RS232/2, RS232/3, and RS232/4. MDOS assumes that AUX means RS232/1 (or just RS232) unless you or an application specify otherwise. On a typical system, RS232/1 could be used for a modem, and RS232/2 for a serial printer.

MDOS reserves these names for devices only; you cannot give any of these names to a file.

CONTROLLING THE DISPLAY

The MODE command has several display related options. Which one you choose depends on the type of display you are using and how much you want to see on the screen.

The MODE command's format to control the display is as follows:

```
MODE [n]
```

The first parameter (n) in the MODE command controls whether the display is on 40, 60, 80, or 90 columns across the screen. Normally the display is in 80 columns, to switch to 40 columns you would enter the following command:

```
A>MODE 40
```

After the MODE command is entered MDOS clears the screen and shifts to the proper number of columns.

```
A>MODE 60
```

Activates Graphics Mode SIX.

```
A>MODE 90
```

Activates TEXT Mode II, 26 lines.

The color of the foreground (text) can be controlled by

```
A>MODE Fx
```

and the background by

```
A>MODE Bx
```

Where x in both cases is a color number as follows:

```
Transparent  1      Medium Red     9
Black        2      Light Red     10
Medium Green 3      Dark Yellow   11
Light Green  4      Light Yellow  12
Dark Blue    5      Dark Green    13
Light Blue   6      Magenta       14
Dark Red     7      Gray          15
Cyan         8      White         16
```

CONTROLLING THE PRINTER

The printer normally prints a maximum of 80 characters per line, and six lines per inch. In can also print smaller type, called condensed type. That fits 132 characters on a line. This ability to change line widths is often useful for printing spreadsheets and other documents wider than 80 characters. The printer can also print eight lines per inch, to fit more lines on a page.

The format of the MODE command to control the printer is as follows:

```
MODE PIO[/#]:[n][,m]
```

When specifying which printer is to be configured, PIO must be used with the number (#) of the port. For example, the first PIO port would be specified as PIO/1 (although just PIO can be used to specify the first printer port.)

The second parameter (n) in the Mode command controls how many characters per line, either 80 or 132, are to be printed. Normally the printer width is set to 80 characters. To switch it to 132 characters per line you would enter the following command:

```
A>MODE PIO:132
```

MDOS would reply:

```
PIO/1: set for 132
```

The third parameter (m) controls how many lines per inch, either 6 or 8, is to be printed. Normally the printer is set to 6 lines per inch, however if you want to switch to 8 lines per inch you would enter the following command:

```
A>MODE PIO:80,8
```

MDOS would reply:

```
PIO/1: set for 80
Printer 6 lines per inch set
```

A printer name must always be included. If you omit the width (n), MDOS leaves the current width unchanged, but a space and the comma must be included so MDOS understands that you have omitted the width.

CONTROLLING THE SERIAL COMMUNICATIONS PORT

Serial communications is controlled by several characteristics, or parameters, that define how fast and in what form data is transmitted. Different devices often require different parameter settings. The communications parameters of your serial port must match those of the device with which you want to communicate. Before you can use the communications port, you must set these parameters with the MODE commands, unless an application program does it for you.

The communications parameters you can set include:

- Baud - speed in characters per second that are sent or received. 110, 300, 600, 1200, 2400, 4800, and 9600 are valid settings. MOS does not assume a default value.
- Parity - the kind of error checking technique used. None (N) Odd (O), and Even (E) are valid settings. MDOS assumes Even (E) as a default value.
- Databits - the number of electrical signals required to define a character. 7 or 8 are valid settings. MDOS assumes 7 as a default value.
- Stopbits - the number of electrical signals that mark the end of a character. 1 or 2 are valid settings. MDOS assumes 2 if the baud rate is 110, otherwise 1 as the default value.

When used to initialize a serial communications port, the MODE command uses these parameters:

```
MODE RS232[/n]:baud[,parity[,databits[,stopbits]]]
```

When specifying which serial port is to be configured, RS232 must be used with the number (n) of the port. For example, the first RS232 port would be specified as RS232/1 (although just RS232 can be used to specify the first serial port.)

A value must be specified for the baud rate each time this MODE command is entered. MDOS assumes the default values for the other parameters unless they are specifically changed. If any parameter is omitted from the MODE command, the comma that precedes it must still be included to show MDOS that parameter was omitted.

Examples: Controlling the Serial Communications Port

To set the baud rate for RS232/1 to 1200 and accept the default values for the other parameters, the following command would be entered:

```
A>MODE RS232:1200
```

MDOS replies by reporting the current setting of each parameter:

```
RS232/1: 1200,e,7,1
```

The report shows that the baud rate is set to 1200, parity is even, databits is 7, and stopbits is 1.

To set the baud rate for RS232/2 to 300, parity to odd, leave databits set to 7, and set stopbits to 2, you would enter the following command:

```
A>MODE RS232/2:300,o,,2
```

MDOS confirms the setting by responding:

```
RS232/2:300,o,7,2
```

CONNECTING A SERIAL PRINTER

If you want to use a serial printer attached to a communications port, you must use the MODE command to tell MDOS to send printer output to the communications port instead of the regular parallel (PIO) printer port. Before the printer output is redirected, you must first set the parameters of the serial communications port to the values required by the printer.

When used to direct printer output to a serial communications port, the MODE command has the following format:

```
MODE PIO/#:=RS232n:
```

PIO/# is the name of the printer port (PIO/1 or PIO/2), whose output is to be redirected. RS232n is the name of the serial communications port (RS232/1, RS232/2, RS232/3 or RS232/4.) Both parameters must be entered.

Example: Connecting a Serial Printer

To redirect output from PIO/1 to serial port RS232/1, the serial port must first be set to match the communications parameters of your printer, then you would enter the following command:

```
A>MODE PIO/1:=RS232/1:
```

MDOS would acknowledge:

```
PIO/1: redirected to RS232/1
```

Now all output that would normally go to PIO/1 would be sent to RS232/1 instead. To cancel the redirection and restore the printer output to PIO/1, the following command would be entered:

```
A>MODE PIO/1:1,1
```

The two numbers following PIO/1 correspond to the digit in the name of the communications port (RS232/1) and printer (PIO/1), respectively. If you had redirected PIO/1 to RS232/2, you would cancel the redirection by entering

```
MODE PIO/1:2,1.
```

COPYING FROM A DEVICE TO A FILE OR ANOTHER DEVICE

The COPY command can be used to copy from a device to a file. This technique can be used to create short text files by copying from the keyboard to a file.

You can also copy from one device to another. Copying from the keyboard to the printer, for example, is a quick and convenient way to print short notes or lists.

When using the COPY command to copy from a device to a file, or to another device, the following format is used:

```
COPY [source] [target]
```

The source is the device to be copied from, and the target is the file or the other device to be copied to. If you are copying from a device to a file, normal file parameters, such as drive letter and pathname, are in effect. When you copy from one device to a file or another device, MDOS continues to copy until it comes to the character (Ctrl-Z) that marks the end of a file. When you copy from the keyboard, you can send the end-of-file character by pressing Ctrl-Z as well.

Example: Copying From a Device to a File or Another Device

To copy from the keyboard (CON) to the printer (PRN), the following command would be used:

```
A>COPY CON PRN
```

Now everything you type is displayed on the screen and sent to the printer until a Ctrl-Z is encountered. Suppose you were to type and enter a few lines of text, the following might occur:

A few notes from the keyboard on the screen are being sent to the printer using the COPY command.

```
^Z
1 File(s) copied.
A>_
```

REDIRECTING COMMAND OUTPUT

The result, or output, for most commands is some action, such as copying a file using the Copy command. The output of a few commands, however, such as DIRectory, CHKDSK, and TREE, is a report. Usually these reports are sent to the display; MDOS has sent them to the standard output device, the console. MDOS will also let you send reports and other output to some other device, such as a printer, or to a file. To redirect the standard output of a command that sends its output to the display, follow the command name with > and the name of the device or file to which the output is to be sent. This technique makes it easy, for example, to print a copy of the directory or tree structure.

Examples: Redirecting Command Output

To print a copy of the directory, use the command:

```
A>DIR > PRN
```

The > tells MDOS to redirect the output of the Directory command, and PRN tells MDOS where to send it; to the printer.

To directly type a batch file and save it to disk:

```
A>COPY CON B:BATCH
```

The batch file may then be typed ending with a ^Z. It will then be saved to drive B under the filename BATCH.

TAILORING YOUR SYSTEM

Although it may seem that MDOS sometimes offers more options than you need, those options give you flexibility in tailoring MDOS; they let you adapt the computer to yourself rather than adapt yourself to the computer.

CHANGING THE SYSTEM CONFIGURATION

Unlike other MDOS commands which tell MDOS what to do, configuration commands tell MDOS how to do something. These commands are required to initially set up how your computer uses its memory and devices.

Usually commands are entered at the system prompt, but configuration commands are put into a special system configuration file. This file is always named AUTOEXEC and must be in the root directory of your MDOS disk. MDOS carries out these special commands only when MDOS is started; if you change a command in the system configuration file, you must restart MDOS for the new command(s) to take effect.

Some application programs or accessory devices require that certain commands be in the system configuration file. The documentation of the application program or device usually includes the instructions you will require to make any necessary adjustments to your system configuration.

If you need to add a configuration command and your MDOS disk already has a file named AUTOEXEC in the root directory, use a text editor, such as MY-WORD in the PE mode to add your configuration command to the file. If your MDOS disk doesn't have an AUTOEXEC file in the root directory, use a text editor to create the file and put your configuration commands in it.

SIMULATING A DISK DRIVE IN MEMORY

MDOS reads from, and writes to, diskettes quickly, and uses fixed disks even more quickly. However both disk drives and fixed disk drives are mechanical and quite slow compared to the computer memory. MDOS lets you set aside a portion of the computer memory for use as a simulated disk drive, making it possible for disk operations to be performed at memory speed. This simulated disk is usually referred to as a RAMdisk, because it acts just like a disk drive and resides in the computer's random access memory (RAM). A RAMdisk behaves like any other disk. It has an assignable drive letter and a directory, and you can specify it in any command that uses a disk. It is much faster than a real disk drive, and the difference is especially noticeable when you use commands such as Copy that work with disk drives, or when you use application programs that access the disk frequently. To use the RAMdisk drive, you would copy the files you need to the RAMdisk after MDOS has started.

MDOS automatically assigns DSK5. to the RAMdisk. You must use the ASSIGN command (see later) to assign a drive letter to it.

Although a real disk drive has a fixed capacity, a RAMdisk can have whatever capacity you want, within certain limits. At no time can you specify a RAMdisk to be larger than total amount of available memory. Unfortunately, setting up a RAMdisk in memory has some drawbacks. The memory used for a RAMdisk reduces the amount of memory available to programs, and the contents of the RAMdisk are lost each time you turn the computer off. When you set up a RAMdisk, make sure you leave enough memory for your programs to use, and be sure to copy any files created or altered on the RAMdisk onto a real disk before turning the computer off.

The format of the RAMDISK configuration command is as follows:

```
RAMDISK [size]
```

The size parameter designates the size of the RAMdisk in kilobytes. The maximum value allowed for the RAMdisk command is the total available memory of your computer.

Example: Simulating a Disk Drive in Memory

Suppose your computer has two disk drives and 512K of memory, and you would like to set up a RAMdisk to store your word processing files. You decide that you can afford to use 180K of memory for the RAMdisk. First, using TI-Writer or another text editor, you would put the following RAMdisk configuration command in the AUTOEXEC file:

```
RAMDISK 180
```

When you restarted your system MDOS would allocate 180 Kbytes of memory for use as a RAMdisk, which could be used as if it were an actual disk drive.

The RAMDISK will be retained whenever going back and forth between the 9640 mode and the TIMODE if the same size RAMDISK is retained and a cold boot is not performed.

DESIGNATING STORAGE DEVICES LOCATIONS AS DISK DRIVES

Floppy drives, hard drives, and RAMdrive cards may be configured into the system as DSK0 through DSK9, in fact to use anything other than floppy drives it is necessary to use the REMAP command in a batch file, usually AUTOEXEC.

The following drive types are now available at the following character definitions (Rave RAMdisk code has not been tested):

```
Drive Alpha.    Device
Assignment
_____________________________________________________________________
A    Floppy #1 (Any floppy only controller)
B    Floppy #2 (Any floppy only controller)
C    Floppy #3 (Any floppy only controller)
D    Floppy #4 (Any floppy only controller)
               (that supports 4 floppies)
E    Internal Geneve RAMdisk     800K
F    Internal Horizon RAMdisk 8 bit CRU >1400 256K
G    Internal Horizon RAMdisk 8 bit CRU >1600 256K
H    Internal Rave RAMisk
I    HFDC Emulate File       800K
J    HFDC Floppy #1
K    HFDC Floppy #2
L    HFDC Floppy #3
M    HFDC Floppy #4
N    Internal Horizon RAMDISK 16 bit CRU >1400 3.2M
O    Internal Horizon RAMDISK 16 bit CRU >1600 3.2M
P    Internal Horizon RAMDISK 16 bit CRU >1000 3.2M
Q    Internal Horizon RAMDISK 16 bit CRU >1700 3.2M
R    Internal Horizon RAMDISK 16 bit CRU >1800 3.2M
S    Internal Horizon RAMDISK 16 bit CRU >1900 3.2M
T    Reserved for Future Expansion
U    Reserved for Future Expansion
V    Reserved for Future Expansion
W    Reserved for Future Expansion
```

The following drive slots are available for remapping:

Slot	Usage
DSK0	GPL Emulate file, any device. If not assigned, no drives are shifted down by 1 If assigned, all drives are shifted by 1 and lose DSK9.
DSK1	Floppy "slot" 1, typical A:
DSK2	Floppy "slot" 2, typical B:
DSK3	Floppy "slot" 3, typical C:
DSK4	Floppy "slot" 4, typical D:
DSK5	Floppy "slot" 5
DSK6	Floppy "slot" 6
DSK7	Floppy "slot" 7
DSK8	Floppy "slot" 8
DSK9	Floppy "slot" 9

How to use the REMAP Command:

```
REMAP [slot][drive type]
```

[slot] uses options 0,1,2,3,4,5,6,7,8,9
[drive type] uses options of A to W.

Note: There must be NO spaces between the values and only 1 space following the REMAP command.

REMAP is effective when using GPL unless using code that accesses the device at hardware level [i.e. track copiers, etc., or probably using ROMPAGE].

Examples

```
REMAP 9I
```

- remaps HFDC Emulate file (if available) to DSK9

```
REMAP 4A
```

- remaps Floppy #4 to slot 1 to answer as DSK1

```
REMAP 0F
```

- remaps Horizon RAMdisk 8 bit CRU >1400 to GPL emulate file while using EXEC/GPL

PRINTING WHILE PERFORMING OTHER TASKS

Suppose you wanted to print a rather large letter that you just typed in using your word processor. Normally, you would have to wait until the entire letter was printed before you could type in another letter, or use the computer for some other task. MDOS lets you set aside a portion of the computer's memory for use as a print spooler, making it possible to print a file while you concurrently perform other tasks. When the print spooler is invoked, the file you wish to be printed is sent to the portion of memory assigned for print spooling. This memory acts like a holding tank. Parts of the file are sent, or spooled, to the printer when the computer is not being used to its full processing potential. This allows you to use the computer for other work while the file is printing.

The print spooler is an output device only. It can be invoked using its own set of special device names. You can address the print spooler as SPPRN, which is just an extension of the Printer device name. To have output sent to the printer through the print spooler specify SPPIO/1 or SPPIO/2, which corresponds to PIO/1 or PIO/2 respectively. Thus, the printer can be addressed normally, or through the print spooler. MDOS assumes that SPPRN means SPPIO/1 (or just SPPIO) unless you or an application specify otherwise. Just like with the RAMdisk, memory used for the print spooler reduces the amount of memory available to programs. The format of the print spooler configuration command is as follows:

```
SPOOL [size]
```

The size parameter designates the size of the print spooler in kilobytes. The maximum value allowed for the Spool command is the total available memory of your computer.

Example: Setting Up a Print Spooler in Memory

Before you can access a print spooler, memory must be allocated, or assigned, to be used for print spooling. You decide that you can afford to use 64K of memory for the print spooler. First, using MY-WORD or another text editor, you would put the following print spooler configuration command in the AUTOEXEC file:

```
SPOOL 64
```

When you restarted your system, MDOS would assign 64 Kbytes of memory for use as a print spooler, and would activate the necessary print spooler device names.

USING THE COMPUTER IN TI-99/4A MODE

Almost all disk-based applications (including solid state Command Modules converted to disk and previously used with the Texas Instruments TI-99/4A Home Computer), can also be used with the MYARC 9640 Computer. MDOS lets you set aside a portion of the computer's memory to be used exclusively for TI-99/4A applications. This segment of memory will actually imitate the entire operating environment of the TI-99/4A computer and allow you to utilize programs originally developed for the 99/4A. Using such TI-99/4A programs may require two minor adjustments. (See also "Controlling System Execution Speed" below.)

A full 96 Kbytes of memory is permanently set aside when the TIMODE command is encountered in AUTOEXEC file. Even if you are not using a TI-99/4A application, this memory is not available for RAMdisk, print spooler, or 9640 based applications. This memory can only be regained when the computer is restarted and the TI Mode configuration command is not encountered in the AUTOEXEC file.

The format of the TI Mode configuration command is as follows:

```
TIMODE
```

There are no parameters for the TI Mode command. However, it must appear before any other configuration command in the AUTOEXEC file in order to enter the GPL mode.

Example: Allocating Your Computer's Memory

Suppose your computer has a total of 512K of memory, two disk drives (A and B), and you want to set up a RAMdisk, print spooler, and have the ability to use TI-99/4A programs. You have decided that you can afford to use 180K of memory for a RAMdisk, and 64K of memory for a print spooler. Using MY-WORD, or another text editor, you would put the following configuration commands in the AUTOEXEC file:

```
TIMODE
RAMDISK 180
SPOOL 64
```

When the system is restarted, MDOS would assign 128 Kbytes of memory for TI-99/4A applications, 180 Kbytes of memory for use as a RAMdisk which is designated as drive C, and 64 Kbytes of memory for print spooling.

DEFINING WORK AREAS IN MEMORY (N/A in V 2.00 or later)

The BUFFERS configuration command defines the number of work areas in memory (buffers) that MDOS uses to handle reading from, and writing to, a disk. Unless otherwise instructed, MDOS uses two buffers. The format of the BUFFERS configuration command is as follows:

```
BUFFERS = [number]
```

The number parameter can be any integer value ranging from 1 to 99.

SPECIFYING THE NUMBER OF FILES MDOS CAN USE (N/A in V 2.00 or later)

The FILES configuration command tells MDOS how many files it can use at one time. Unless otherwise instructed, MDOS can use a maximum of eight files (default) at a time.

The format of the FILES configuration command is as follows:

```
FILES = [number]
```

The number parameter can be any integer value ranging from 1 to 20.

DEFINING THE HIGHEST SYSTEM DRIVE NUMBER

The LASTDRIVE configuration command specifies the highest drive letter that MDOS recognizes as valid. If the system configuration file does not contain a LASTDRIVE command, the highest value MDOS recognizes is G. This command is usually used to specify a higher letter (up to Z) if eight or more drive letters are needed. Use this command after REMAP if used. The format of the LASTDRIVE configuration command is as follows:

```
LASTDRIVE = [letter]
```

MISCELLANEOUS COMMANDS FOR OCCASIONAL USE

There are a few commands you might occasionally need for convenience. These commands are also used in tailoring your system to meet your specific needs or applications.

CHANGING THE SYSTEM PROMPT

You can change the system prompt with the PROMPT command to display more, or less, than the current drive letter. The change takes effect as soon as the command is entered.
The PROMPT command's format is as follows:

```
PROMPT [prompt-text]
```

The prompt text defines the new system prompt. You can use any characters you wish. You can also cause the new prompt to include one or more items of useful information by including a dollar sign followed by one of the following characters, to specify what you want the prompt to contain:

Character	Produces
d	The current date
t	The current time
p	The current directory
n	The current drive
v	The version number of MDOS
g	A greater-than sign (>)
l	A less-than sign (<)
b	A space character ("^")
q	An equal sign (=)
$	A dollar sign ($)
h	A backspace and erasure of previous character
_	A signal to end the current line and start a new one (the character is an underscore)

You can include as many combinations of $, followed by a character, as you wish. MDOS ignores any combination of a $ followed by a character not in the preceding list. If you enter the PROMPT command without any parameters, MDOS indicates the present prompt setting. For example, if the prompt is A>, MDOS returns ng.

Examples: Setting the System Prompt

To define the system prompt as "Command:", type and enter the following:

```
A>PROMPT Command:
```

MDOS will display the new system prompt as follows:

```
Command: _
```

To define the system prompt as the current drive and directory, type and enter:

```
A>PROMPT $p
```

MDOS would display the new system prompt as follows (if the current directory was \office):

```
A:\office _
```

To define the system prompt as the current date, time, and current drive with a greater-than sign afterwards, type and enter:

```
A>PROMPT $_$d$b$b$t$_$_$n$g
```

MDOS would display the new system prompt as follows:

```
06-26-90 11:17:25
A>
```

ASSIGNING A DRIVE LETTER TO A DIFFERENT DRIVE

The ASSIGN command causes MDOS to route disk input/output requests for one drive into disk input/output requests for another drive. Because the ASSIGN command affects all requests for a drive, you should use it with some caution.

The ASSIGN command's format is as follows:

```
ASSIGN x=y:
```

The first drive letter (x) is the letter to be assigned to a different drive, while the second drive letter (y) is the letter to be used in place of the first. It may also be DSKn. If you omit both drive letters, MDOS lists the present assignments in effect.

For example, suppose you have a graphics program that requires that all data files be on drive B, but you want to use your fixed disk (drive E) for data files. To tell MDOS to assign all requests for drive B to be routed to drive E instead, type and enter the following command:

```
E>ASSIGN B=E:
```

You could have MDOS assign all requests for drive A and B to be routed to drive E also. This could be done using the following command:

```
E>ASSIGN A=E: B=E:
```

The assignment remains in effect until you restart MDOS, or cancel the assignment by typing:

```
>ASSIGN A=A: B=B:
```

ASSIGNING THE RAMdisk A DIFFERENT DRIVE LETTER

When memory is first allocated to be used as a RAMdisk, MDOS automatically assigns DSK5 to the RAMdisk. You must use the ASSIGN command to set the drive letter. There may be times you would like to change the drive letter designation of the RAMdisk; this can be accomplished using the Assign command in the same manner it was used with physical disk drives.

When the RAMdisk is assigned a new drive letter, all requests for the RAMdisk must be done using the new drive letter. If the RAMdisk is assigned the letter of a disk drive that is present in the system, all requests for that disk drive would be routed to the RAMdisk and you would be unable to access the physical disk drive.

For example, suppose your system consisted of two disk drives (A and B) and you had allocated memory for use as a RAMdisk. MDOS automatically assigns the DSK5 to the RAMdisk. If the following command was executed, the RAMdisk would now be accessed as drive A, and the physical disk drive could no longer be accessed.

```
B>ASSIGN A=DSK5:
```

To regain the use of the physical disk drive (A), the RAMdisk would have to be assigned a new letter.

DISPLAYING THE MDOS VERSION NUMBER

The VERsion command displays the number of the version of MDOS you're using. If you use more than one version, or if you are using someone else's machine, this gives you a quick way to check the version. The VERsion command's format is as follows:

```
VER
```

The VERsion command has no parameters.

CHANGING COMMAND LINE INTERPRETATION

Normally lowercase is converted to uppercase when a command line is entered. If you need lowercase filename or the like passed through use the command:

```
CASE ON
```

Returning to the normal conversion can be accomplished by:

```
CASE OFF (Default)
```

ADDITIONAL FLOPPY DISK CONTROL

Normally MDOS verifies disk writes. The 99/4A did not. Using the following command will result in speedier floppy access:

```
VERIFY OFF
```

Returning to verified writes is done with:

```
VERIFY ON
```

VERIFY alone will return ON/OFF status.

CONTROLLING FLOPPY DISK HEAD STEP TIME

On faster floppy drives the head step time can usually be reduced with a command:

```
SETDSK [drive number][head step][tracks]
```

[drive number] is from 1 to 4 and is valid ONLY for floppy only controllers.

[head step] is from 0 to 3. Vary settings, "usually" 0 for optimal performance, but depends upon floppy type. 0 is the fastest.

[tracks] valid numbers are 4 or 8 to indicate 40 or 80 tracks respectively.

Examples:

```
SETDSK 104 sets drive 1, head step 0, 40 tracks

SETDSK 228 sets drive 2, head step 2, 80 tracks

SETDSK 308 sets drive 3, head step 0, 80 tracks
```

Leave only one space between command and options and do not separate the numbers or an error will occur.

APPENDIX: A

SUMMARY OF MDOS COMMANDS

Note: DSKn. may substitute for d: in many cases.

`ASSIGN [x=y]:`
Assigns requests for one disk drive to another disk drive.

`ATTRIB [+/-P][d:][filename]`
Sets or displays the read only status of a file or set of files.

`CASE [OFF/ON]`
Controls the conversion of LC to UC in the CLI.

`CD [[d:]path] or CD[[d:]path]`
`CHDIR[[d:]path] or CD[[d:]path]`
Changes the current working directory.

`CHKDSK [d:][filename][F]`
Checks a disk and reports status of the disk and the computer's memory.

`CLS`
Clears the screen.

`COPY [d:][filename] [d:][filename]`
Copies a file to another file. CON and PRN can be used.

`DATE`
Sets and displays the current date.

`DIR [d:][filename][/W][/P]`
Displays the files on a disk.

`DISK1  DISK1 [ON/OFF]`
Controls HDS1.DSK1. emulation.

`DISKCOMP DISKCOMP [d:][d:]`
Compares two diskettes sector by sector.

`DISKCOPY DISKCOPY [d:][d:]`
Copies a complete diskette to another diskette.

`DEL [d:][filename]`
`ERASE [d:][filename]`
Deletes files.

`FORMAT [d:][/1][/16][/18][/36][/80][/N][/V]`
Formats a diskette for use.

`HARD [OFF/ON]`
Controls HFDC access.

`LABEL [d:][volume label]`
Assigns, changes, or deletes name of a diskette or disk.

`MD [d:]path`
`MKDIR [d:]path`
Creates a subdirectory.

`MODE [n]`
Selects the number of characters in the active display.

`MODE [B/F][N]`
Sets background and foreground colors.

`MODE PIO [/#]:[n][m]`
Controls the line width and spacing of the printer.

`MODE RS232[/n]:baud[,parity][,databits][stopbits]`
Controls the communication parameters of a serial port.

`PATH [[d:]path[;[d:]path ....]]`
Designates where MDOS will search for commands within a directory structure.

`PROMPT [prompt-text]`
Changes system prompt.

`REMAP [slot][drive-type]`
Configures the location of various storage devices..

`RENAME [d:][filename] [filename]`
`REN [d:][filename] [filename]`
Changes the name of a file

`RD [d:] path`
`RMDIR [d:] path`
Removes a subdirectory

`SETDSK [drive-no.][head-step][tracks]`
Set floppy drive headstep.

`TI`
Returns status, see Autoexec commands.

`TIME`
Set and displays the current time.

`TREE [d:]`
Displays the path of each subdirectory on a disk.

`TYPE [d:][filename][/M]`
Displays the contents of a text file.

`VER`
Displays the version number of MDOS.

`VERIFY [OFF/ON]`
Controls verification during disk writes.

`VOL [d:]`
Displays the volume label of a disk.

The following are batch file commands are used only in BATCH and AUTOEXEC files:

`ECHO [ON/OFF][message]`
55 Controls whether the commands are displayed as they are executed.

`FOR %%variable IN (set) DO command`
Carries out a command for each value in a given set, and a value is assigned to each value in the given set.

`GOTO label`
Transfers control to the line following :label.

`IF [NOT] condition command`
Conditional execution of commands.

`PAUSE [message]`
Causes the system to pause until a key is pressed.

`REM [message]`
Displays a message if ECHO is ON.

The following are used only in AUTOEXEC files:

`BUFFERS = [number] (Obsolete command)`
Defines the number of disk buffers MDOS allocates memory for.

`FILES = [number]  (Obsolete command)`
Defines the number of files MDOS can us at one time.

`LASTDRIVE = [letter]`
Specifies the last valid drive letter.

`MIRROR [1/2]`
Stores an extra bitmap of cylinder 0.

`RAMDISK [size]`
Specifies how much memory will be allocated for use as a RAMdisk.

`SPOOL [size]`
Specifies how much memory will be used as a print spooler.

`TI [ON/OFF]`
Controls WDS emulation in GPL mode.

`TIMODE`
Allocates 96K of memory for use with TI-99/4A-based application programs.

APPENDIX: B

ADDITIONAL LIMITATIONS FOR USE WITH DIR COMMAND

When listing a directory with the DIR command additional limits may be made to the files to be listed by use of the following: Note, the free sector count will not be correct. This also works with COPY, DELETE, TREE, and TYPE.

Use the command >DIR `k

Where k is in the following list with its meaning:

`D D/V or D/F files only listed
`I I/V or I/F
`DV D/V
`DF D/F
`IV I/V
`DV163 D/V 163
`IF128 I/F 128
`P Programs
`V Variable
`F Fixed
`Dir Directory

APPENDIX: C

1.44 MB FLOPPY DISK SUPPORT

This support is only available in MDOS versions 1.60 or higher. Versions less than 1.60 do not include the support. If you have a 1.44MB floppy and wish to format it to it's highest density, you can issue the following command from the MDOS prompt to format your disk to maximum capacity.

FORMAT A: /2 /36 /80

Where A: is your drive letter
Where /2 is the number of sides
Where /36 is the number of sectors/track
Where /80 is the number of tracks

Please remember that the Myarc HFDC is the only floppy controller capable of formatting diskettes in this format.

Please also remember that your Myarc HFDC controller contains dip switches that you must use to configure your drives. This is how MDOS knows whether you have a 5.25" drive, a 720K drive, or a 1.44MB drive. Once your disk is formatted, most disk managers will not work with this high density and will report incorrect information. The Myarc HFDC Disk Manger V will support this structure and so will Clint Pulley's Directory Manger. I recommend Directory Manager for using these new formats.

Many people may remember difficulties of using CorComp formatted disks with the Myarc HFDC. I "believe" these difficulties have now been solved with this release of MDOS. I do not know if you will have difficulties reading disks formatted earlier. If a disk will not read with the Myarc HFDC and was previously formatted with a CorComp disk controller, you may need to use your CorComp disk controller to read the disk.

Jim has also solved a problem that many users saw while formatting a floppy disk when using the Myarc HFDC. Many users had to reformat the disk a second time as the disk would not reformat. This problem has been corrected in the FORMAT command by attempting to read the first sector of the disk whether it was previously formatted or not. For reasons I won't explain, the controller chip on the HFDC "knows" what kind of disk it is dealing with.

A 9216B chip is required on the HDFC for reliable 1.44MB usage.

FORMATTING RAM DISKS

A command FORMAT n:/Kxx allows formatting a RAMdisk to xx kilobytes. /N and /V may follow if wanted.

READ THIS NOTICE!!!!!! VERY IMPORTANT!!!!!!!!

There are also some things you may need to consider when/if you switch your drive controllers or your drive configuration. Be very careful on the use of terminal resistor packs on your drive. If the terminal resistor packs are improperly used, you will be UNSUCCESSFUL in formatting floppies. I can not over emphasize the PROPER use of resistor packs with the Myarc HFDC. What you previously used with other controllers because it "worked" (as it did for me), may not work with the Myarc HFDC. The proper use is as such. The drive with the LONGEST cable, not necessarily the last drive, should be the only drive with a resistor pack. Be careful using 1.44MB floppy drives and their configuration. Some 1.44MB floppy drives have dips to disable the resistor pack, some have removable resistor packs, and some may not have anything to enable or disable the resistor packs at all. Much testing may be required to properly configure your 1.44MB drive AND your other drives.

AND FINALLY, ONE MORE MAJOR THING!!!!!

In order to use reliably the High Density support of the Myarc HFDC, you must have a chip on the HFDC with the following designation - 9216B -. Myarc shipped many HFDC's with a -9216-. The 9216 is not rated for the faster speed required for high density reads/writes/formats. There is an outside chance you might have a 9216 that can perform at 9216B speeds, but be VERY careful. Some days it may work then other days it may not.

Cecure Electronics sells an upgrade if you want to use high density.

NOTES REGARDING HIGH DENSITY FLOPPY DRIVES

There have been several users that have tested problems using Mitsumi 1.44MB High Density drives. I highly recommend that if you upgrade to High Density 3.5" drives, USE TEAC drives or you may experience problems that will not allow you to properly format your hard drive.

APPENDIX: D

HARD DRIVE SUPPORT INFORMATION

SUPPORT TO RECOVER DATA AFTER CYLINDER 0 CRASH

This command must be used in an AUTOEXEC file for use with hard drives. The command provides support to recover hard drive data in the event a cylinder 0 or bitmap crash occurs.

The use of the command is as follows:

`MIRROR 1`
Stores bitmap of hard drive #1

`MIRROR 2`
Stores bitmap of hard drive #2

Note: MIRROR 3 is not available as the HFDC hard drive #3 permits reading from that drive only.

This command copies sectors >00->1F to sectors >20->3F of the selected hard drive. In the event of a cylinder 0 failure on your hard drive, then one may use a sector editor (SECTORONE for MDOS) to copy sectors >20 to >3F back to >00 to >1F returning the system to the last successful use of MIRROR.

MDM5 permits formatting the hard drive using 32 sectors per cylinder. CFORM has extended the capabilities of the HFDC to make use of 33/34 sectors per cylinder. In the event you choose to format 33/34 sectors per cylinder with CFORM (instead of 32), then reformatting cylinder 0 of your drive in the event of a failure will invalidate the use of MIRROR.

If you reliably use MIRROR and format your hard drive to 32 sectors per cylinder, your chances of recovering data from cylinder 0 failures are much greater. Following a cylinder 0 failure, you MUST only format ONE cylinder. Do not format the entire hard drive or you will overwrite the information that MIRROR stored. Following the restoration of the backup sectors (copying 20->35 to >00->1F), it is possible that a DIR command will abort if files were deleted or if illegal filenames now exist. In they event that they do, you should still be able to use the CHDIR command to access other subdirectories. Access to the root directory and other subdirectories depends upon the deletions or modifications of files that took place following the last use of MIRROR. After having restored the hard drive and recovering any files that you need to recover, it is HIGHLY recommended that you then do a complete reformat of your hard drive. Do not ever save any files to the hard drive following a restoration. You further risk corrupting additional bitmaps.

CONTROL OF HDFC ACCESS

The commands for controlling HDFC access are:

`HARD OFF`
turns off access

`HARD ON`
turns on access, default

WDS SUPPORT IN GPL

These modifications are for GPL mode only and REQUIRE V2.0 GPL or later.

A new command has been added to either permit one to use the HFDC as WDS devices under GPL mode, or to continue using it as the recent versions of MDOS (0.97H, 1.21H, 1.23H) with GPL device names as HDS. These modifications do NOT require the use of OLDDSR/ROMPAGE while in GPL mode. Any device with WDS in the DSR (Myarc HFDC, Bud Mill's soon to be SCSI) can be toggled on if the card is at CRU >1200. ONLY devices with CRU dipswitches set to CRU >1200 can operate as WDS.

If you have a device at CRU >1200 and you want to use it as WDS under GPL mode, you MUST have the following switch set in your AUTOEXEC. The switch will NOT work in a batch file.

Usage required in Autoexec.

`TI ON`
turns on WDS emulation

`TI OFF`
default status

Usage not required in Autoexec, but will return status display.

`TI`
identifies WDS emulation on

Users wishing to use the WDS ON capability as described must either use just a Myarc HFDC as their floppy controller, or use a CorComp Disk Controller. Attempts to use WDS at CRU >1200 on the Myarc HFDC with a TI or Myarc Disk controller present will cause lockups. The TI and Myarc Disk controller card DSR's would require modification to allow this support.

Using the HDFC or other controller card, you presently have 3 CRU addresses you may boot from due to limitations posed by LOAD/SYS and the boot eprom. The following restrictions for usage are described below (CRU's >1000, >1100, >1200).

OPTION

```
@CRU >1000
```

System: HFDC at CRU >1000 and disk controller at CRU >1100

1. TI OFF No support for WDS usage. Full use of the DSK1 emulation, DSK1 directory, and DSK directory emulation are provided. HFDC is accessed as HDS.

2. TI ON Will lockup as no WDS device available and you confused the system upon entry into GPL mode. Modify AUTOEXEC to reflect TI OFF.

```
@CRU >1000
```

System HFDC at CRU >1000, disk controller at CRU >1100, SCSI (or HFDC???) at CRU >1200

3. TI OFF No support for the card at CRU >1200, but the HFDC will respond as HDS. DSK1 emulation, DSK1 subdirectory and DSK emulation will work on the HFDC. Device names will be HDS.

4. TI ON The HFDC at CRU >1000 will respond as HDS and the SCSI (and possibly second HFDC in the system) will respond as WDS. DSK1 emulation, DSK1 subdirectory and DSK emulation will work on the HFDC but will not be accessible on the controller card at CRU >1200.

```
@CRU >1100
```

System HFDC at CRU >1100, no other cards.

5. TI OFF DSK1 emulation, DSK1 subdirectory, and DSK emulation will work on the HFDC. All device names will be HDS.

6. TI ON Will lockup as no WDS device available and you confused the system upon entry into GPL mode. Modify AUTOEXEC to reflect TI OFF.

```
@CRU >1100
```

SYSTEM HFDC at CRU >1100 and a second card at CRU >1200

7. TI OFF DSK1 emulation, DSK1 subdirectory, and DSK emulation will work on the HFDC. All device names will be HDS.

8. TI ON The HFDC at CRU >1100 will respond as HDS and the SCSI (and possibly second HFDC in the system) will respond as WDS. DSK1 emulation, DSK1 subdirectory and DSK emulation will work on the HFDC but will not be accessible on the controller card at CRU >1200.

```
@CRU >1100
```

Disk Controller at CRU >1100 and HFDC/SCSI at CRU >1200

9. TI OFF HFDC will only respond as HDS and will not support DSK1 emulation, DSK1 subdirectory or DSK emulation on the HFDC.

10. TI ON HFDC will respond only as WDS and will not support DSK1 emulation, DSK1 subdirectory or DSK emulation on the HFDC.

NOW WHAT DOES THIS MEAN?

What this all means is if you use two controller cards (HFDC/Floppy) and select OPTION #10, programs like the V1.50 MDM5, Gentri, HardMaster and other programs using WDS will all work properly with only one restriction caused by a bug of the HFDC DSR.

If you use the HFDC at CRU >1200 with TI ON, do not use any GPL based program to Make or Remove directories such as MDM5. Create your directories from MDOS mode. MDOS mode and the DSR do not like one another and will cause MAJOR FILE LOSS problems if you CREATE a DIRECTORY with MDM5 and then try to remove it from MDOS prompt (or vice versa). DON"T DO IT!!!! TI-99/4A people can do it as the one bug did not care about the other bug, but MDOS cares.

As of this release, moving the HFDC to CRU >1200 while using TI ON will cause problems with EXEC. Barry Boone will be provided the necessary information so that a new release of EXEC can accommodate these new features of MDOS with the additional GPL >2x support.

When using the HFDC as WDS in GPL mode, do not EVER call for a file from HDS. Attempting to do so will cause unpredictable results as the operating system is trying to access the HDS device, however it did not know that WDS has changed the parameters by using the DSR. You will probably need to reconfigure My-Word, Telco, Gentri, and MDM5 to respond as using WDS if you choose the TI ON option.

If you decide to switch from using WDS to HDS or vice versa, edit your AUTOEXEC file and REBOOT your operating system. A warm boot will not reconfigure your system.

In this configuration, the GPL mode support for the SCSI is complete. I know this as I have expanded the use of WDS from WDS1...to WDS8. WDS1 and WDS2 have been tested on the HFDC.

CONTROLLING HDS1.DSK1. EMULATION

The command, DISK1 <OFF> or DISK1 <ON> turns off or on the HDS1.DSK1. emulation, so users may choose whether files are accessed on that subdirectory while in GPL mode.

APPENDIX: E

GPL NOTES

GPL VERSION 2.0 IS FOR MDOS VERSION 2.00.

This version of GPL should work properly with the new combined MDOS for all users, regardless of whether they have a floppy controller, HFDCC, SCSI, or combination thereof.

GPL now clears previously untouched video registers. Included are the color, sprite, and blink tables. Prior versions did not reset these registers, causing strange results in various graphic modes.

New in Version 2.0.

1. CTRL-ALT-DEL is no longer the only exit to MDOS from the GPL loader screen. The user may now press <ESC>ape to exit, avoiding the possibility of resetting the system.

2. The joystick lines were reversed. Joystick #1 now operates as #1, not as #2 - and vice versa.

3. The FIREBUTTONS are now independently processed. Previously there were problems with rapid-fire in certain programs such as Barrage and Tennis.

CONTROLLING SYSTEM EXECUTION SPEED

Many of the programs found in the command modules produced by Texas Instruments were written in TI's Graphics Programming Language (GPL). When these modules are converted to disk, the GPL programs can be used with the 9640, but they may not run correctly with the MYARC 9640 Computer due to the enormous differences in the execution times (speed) between the two computers. Speed problems may be corrected using the Speed options available on the GPL Interpreter Screen. This includes problems with the RAVE SPEECH ADAPTER, a device that allows you to place the innards of a speech synthesizer inside the Peripheral Expansion Box. This is the only way to get speech on the MYARC 9640 FAMILY COMPUTER. To properly access speech in this way, the GPL speed will normally need to be set at "Speed 1" (see below).

Using the F4 key at the GPL Interpreter Screen allows shifting to slower speeds than the default Speed=5 thus controlling the speed that the system executes GPL programs. The Speed can be set at any one of the GPL-speed values listed below:

1 - Normal TI-99/4A in Basic -Slowest
2 - The TI-99/4A in Extended Basic
3 - Approximately 2 times faster than the TI-99/4A.
4 - Approximately 3 times faster than the TI-99/4A.
5 - Approximately 3 1/4 times faster than the TI-99/4A.

ADDITIONAL NOTES

Be sure to clear cartridge space with F3 & Y when loading a new cartridge.

In a AUTOEXEC file a saved cartridge file may be loaded directly in the same command that loads GPL. For example, E>GPL DSK6.TIXB

NOTES FOR GPL VERSION 6.0

(Fixes made by Tim Tesch unless otherwise noted)

USAGE

To use the new GPL, simply copy the supplied GPL over your existing GPL file. No other modification is required.

MODIFICATIONS LIST

16 May 1998

Version change. No modifications made to the program. Only reason for doing this is to make sure people are using the most recent version, and to keep some semblance of order!

30 June 1996

1. Changed the menu around - DSK1 Emulation is now F6

2. "ROMPAGE" may now be activated from the GPL menu screen by pressing F5. GPL will first page out the master DSR then will perform the equivalent function of ROMpage. Once that is done, you are put into GPL and may use the "WDSx" device.

3. Once you SHIFT-SHIFT-CONTROL back to the menu, the master DSR is restored. The ROMpage standalone program did not do this.

4. Besides cleaner entry/exit, you don't have to worry about the CRU address of the HFDC - GPL relies upon MDOS for its information, meaning no need for hex editing.

5. MDM5, SpellIt, and other programs which make use of the "WDSx" name and the ">2x" routines are usable with this new feature.

Sometime after October 31 1994 but before January 1, 1996

Programmers note: The mapper page for GPL task page zero is stored at location >8100. This was done to facilitate paging to Geneve mode for certain applications, such as the S&T BBS Geneve version of assembly which uses the master DSR for all file access. If you would like to learn how to do this please contact me for more information. This modification is also present in the GPL Cheater program.

31 October 1994 <Halloween> (version 2.2)

1. Changed GPL Interpreter screen around a bit. Copyright changed to reflect 9640*News & contributors.

2. Added option to toggle DSK1 subdirectory emulation on hard drive #1 from GPL interpreter screen using F5. (same as DISK1 ON/OFF in MDOS)

3. Changed the GPL Keyscan per Jeff White's modification in MDOS; also added to EXEC 2.11

01 March 1994 (version 2.0)

1. CTRL-ALT-DEL is no longer the only exit to MDOS from the GPL loader screen. The user may now press <ESC>ape to exit to MDOS, avoiding an inadvertent system reset.

2. The joystick lines were reversed. Joystick #1 now operates as #1, not as #2 - and vice versa.

3. The FIREBUTTONS are now independently processed. The GPL status byte was not being correctly set/reset for each joystick. This caused rapid-fire in certain programs such as Barrage and Tennis. Also, with the buttons being checked in this manner, there ought to be less probability that the buttons are missed.

15 January 1994

1. This version of GPL should work properly with the new combined MDOS for all users, regardless of whether they have a floppy controller, HFDCC, SCSI, or combination thereof.

2. Since the fixes we made to the power up routine in MDOS were successful, only minor changes to the GPL power up were necessary.

3. GPL now clears previously untouched video registers. Included are the color, sprite, and blink tables. Prior versions did not reset these registers, causing strange results in various graphic modes.

APPENDIX: F

CHANGES IN MDOS 2, 5, AND 6

MDOS VERSION 2.21, released November 11, 1994

1. MDOS now contains an imbedded CRC. The value is stored in MDOS using the CRaCkit installation program, run by 9640*News before distribution. Using CRaCkit version 2.0, MDOS can be verified internally. If any changes have been made, either by sector editing or errors during a transfer, CRaCkit will inform you of the problem. More information is available in the doc file (included).

2. Memory pages >C0 through >EF will be treated as FAST RAM if available for system usage.

3. The REMAP table has been changed. Please find the REMAP_TXT file which contains a list of the devices.

Device 20 - PFM Flashdisk #1
Device 21 - reserved for Flashdisk #2

4. If you have not obtained EXEC 2.11 from Tim Tesch, it would be wise to obtain this program, which includes enhancements such as rompage support, supercart support, video fixes, and more. Programs such as MDM5 and Spell-It! can be used from MDOS without using the WDS support (TI ON/OFF).

5. RS232 detect routine modified to be DSR independent thanks to Jeff White.

6. Keyscan modification (same one made to GPL and EXEC 2.11)

- COPY command may still exhibit problems; cause still unknown.
- FORMAT command may fail when both a Myarc floppy controller and Myarc HFDC are in the system. If you experience this problem, the best remedy is to turn your system OFF for 20-30 seconds.

v2.00 CRaCkit Value B74C (unsupported in Crackit 2.0)

The value above is the CRC value calculated with Tim Tesch's CRaCkit Utility. If you run CRaCkit and do not observe this same value, you have a modified or corrupted version of MDOS.

Note: There is one "bug" left in MDOS that has been present all the way back to MDOS 1.14 and possibly earlier. MDOS opens/closes files different than the 4A system and returns file parameters back if there is insufficient space for a file. Due to this, few programs perform all the necessary error detection. When copying files, use Clint Pulley's Directory Manager for MDOS mode and you should never experience the problem. Future updates to MDOS will be available on GEnie in the TI and Orphans RT (page 575) and the 9640 News BBS (1-901-368-0112 at 300 to 14.4K baud). MDOS Support headquarters are on GEnie.

Added support for 3.2MB Horizon RAMdisks using HRD formatting program by James Schroeder.

Added support for 1.44MB floppy drives. See attached document.

Added new switch to FORMAT command per wish of Jonathan Leslie. The new switch is /N. The command when /N is used disables the floppy verification of all sectors on the disk. Floppy formatting is MUCH quicker.

Modified Powerup routine while in GPL mode. Elimates lockup when using ROMPAGE/OLDDSR utilities and exit GPL improperly, thanks Tim Tesch

Added command to H version of MDOS per wish of Dan Eicher

HARD OFF Turns off HFDC access
HARD ON Restores HFDC access (default)

The MODES file has been changed. New items are:

1) MODE 60 --> Graphics Mode SIX activated

2) MODE 90 --> TEXT mode II, 26 line mode activated

Scroll-Back Buffer

1. PAGEUP now works properly
2. When finished looking at the buffer, the screen is returned to the previous state.
3. Changes were made to accommodate the MODE 90 command; buffer now can be reviewed in 26-line mode
4. Bug fixed, 40 column scrollback now works properly
5. Graphics mode trap added. Scrollback only works in the TEXT modes.
6. CTRL-C deactivated while in the scroll-back routine
7. All screen reads/writes are TTYOUT independent
8. Tested with VDP wait states on/off - no ill results
9. SCREEN DUMP ADDED - Pressing "P" dumps the current screen to the printer.

MDOS CLI COMMAND TYPE

1. High ASCII is no longer masked - DV80 files with IBM graphics will be displayed; a file to define the high-ascii will be made available.
2. DF128 files are displayed without any MDOS interpretation. PC text files with CR/LF combinations will display properly
3. The MORE function "/M" has been changed.
    a) works in 24 and 26 line modes
    b) the prompt is now “Press any key, (A)bort, or (N)onstop..." and works equally well with DV80 and DF128 files

4. File-closing problems were eliminated. Certain files would exit the TYPE routine, leaving a file open. Enough of them caused system failure.

New Command to CLI

Code for a new command, DISK1 <OFF> or DISK1 <ON> was added. This will effectively turn on or off the HDS1.DSK1. emulation, so users may choose whether files are accessed on that subdirectory while in GPL mode.

v1.53

Note that EXEC will no longer run MY-Word on stock memory systems unless TIMODE is active. If you have extended memory, TIMODE is not necessary. Use the modified MY-Word loader available from Tim Tesch.

v1.52(F/H)

Note: Users wishing to use the WDS ON capability as described later in this article must either use just a Myarc HFDC as their floppy controller, or use a CorComp Disk Controller. Attempts to use WDS at CRU >1200 on the Myarc HFDC with a TI or Myarc Disk controller present will cause lockups. The TI and Myarc Disk controller card DSR's would require modification to allow this support.

Added new MIRROR command that must be used in an AUTOEXEC file that can be used with hard drives. The command provides support to recover hard drive data in the event a cylinder 0 or bitmap crash occurs.

The use of the command is as follows:

`MIRROR 1`
Stores bitmap of hard drive #1

`MIRROR 2`
Stores bitmap of hard drive #2

Note: MIRROR 3 is not available as the HFDC hard drive #3 permits reading from that drive only.

Note 1: This command copies sectors >00->1F to sectors >20->3F of the hard drive. In the event of a cylinder 0 failure on your hard drive, then one may use a sector editor (SECTORONE for MDOS) to copy sectors >20 to >3F back to >00 to >1F returning the system to the last successful use of MIRROR.

Note 2: MDM5 permits formatting the hard drive using 32 sectors per cylinder. CFORM has extended the capabilities of the HFDC to make use of 33/34 sectors per cylinder. In the event you choose to format 33/34 sectors per cylinder with CFORM (instead of 32), then reformatting cylinder 0 of your drive in the event of a failure will invalidate the use of MIRROR.

Note 3: If you reliably use MIRROR and format your hard drive to 32 sectors/cylinder, your chances of recovering data from cylinder 0 failures is much greater. Following a cylinder 0 failure, you MUST only format ONE cylinder. Do not format the entire hard drive or you will overwrite the information that MIRROR stored. Following the restoration of the backup sectors (copying >20->35 to >00->1F), it is possible that a DIR command will abort if files were deleted or if illegal filenames now exist. In they event that they do, you should still be able to use the CHDIR command to access other subdirectories. Access to the root directory and other subdirectories depends upon the deletions or modifications of files that took place following the last use of MIRROR. After having restored the hard drive and recovering any files that you need to recover, it is HIGHLY recommended that you then do a complete reformat of your hard drive. Do not ever save any files to the hard drive following a restoration. You further risk corrupting additional bitmaps.

- Modified the Geneve Internal RAMdisk "RAMDISK" command to be universal in all memory configurations. In a warm boot or cold boot of the operating system, the RAMDISK will reinitialize if TIMODE is enabled. One may disable the TIMODE in a redirected autoexec file ("&filename") if the RAMdisk is kept the same size without losing the disk contents. This was required to solve TIMODE/RAMDISK memory conflicts in systems with no expanded memory on their Geneve.

- Modified Drive decrements in file L8\SECT2-P so that whatever drive assignments you set from AUTOEXEC, they carry forward to GPL mode. Now, GPL mode will not offset drive sequencing as previous versions from 0.97H to 1.50H which were caused when EMULATE drives were active.

- Fixed usage problems a user found with LABEL command. Problem was actually a bug with BREAD/BWRITE on sector 0 i/o buffering as the Geneve (and TI-99/4A) can not determine when a drive door has been opened.

- Modified file redirection to use DV80 files instead of DF1 files. (thanks Clint)

- Removed VIDEO command. Too many users were using VIDEO ON on occasions where MDOS programs were incompatible and they thought MDOS was the problem. Programmers must implement these commands if they want the extra speed.

I would also at this time like to thank the following individuals for their assistance in the MDOS programming effort:

James Schroeder, Clint Pulley, Alan Beard, Barry Boone, John Johnson, Tim Tesch, Jeff White, and finally all the contributors that made the MDOS purchase possible. THANK ALL OF YOU VERY MUCH!!!

Beery Miller / 9640 News

v1.50

**** NOTE: TI ON/OFF command for WDS support discussed in WDS_TXT

- Fixed problem with formatting floppies attached to the HFDC from the CLI.

- Clint has modified hchar/vchar printing routines resulting in screen writes taking 3.5 seconds versus 6.7 seconds in previous graphic screen writes.

- Fixed conflicts the mouse driver and CLI hooks were having as a result of table overwrites when defining character definitions higher than 128.

- JJ found, Clint fixed a bug with character definitions in certain bitmap modes when reading patterns from the screen.

- Modified TIME/DATE function to not query for input if in batch mode by Al Beard

- Re-modified character set to MDOS original by Beery Miller.

- Added CASE ON/OFF, will convert lowercase filename to uppercase filename automatically if name is passed through the parser routine. Default is to accept lowercase filename and convert to uppercase automatically by John Johnson.

- Modified Horizon RAMdisk support to "force" users to use the proper REMAP configuration if they are using a 16 bit RAMdisk with >256K by James Schroeder.

- Clint fixed Paul Charlton's high speed disk I/O transfer routines Use hard driver interlace of 7 for optimum speed. Floppy I/O will also be speeded up under certain conditions.

v1.23

- Unless new bug reports come about, this will be the last 358 sector version of MDOS (F). All subsequent releases will be hard drive compatible and will not fit on a SS/SD disk.

- A major bug using files in update mode has been fixed (thanks Clint Pulley) that were on the HFDC. Now, TI-BASE, FirstBase, Telco's TOS/PHONE files, and others work properly. If your system presently has files that may be "bad", copying to Geneve Internal RAMdisk or other floppy device with Clint's Directory Manager will prompt you to repair any files that are bad.

Symptoms included opening a database file, accessing a record partial way but not at the end. When the file was closed, it would close wherever the last record was accessed losing everything beyond that point. This bug existed in only the H version of MDOS when working with files stored on the hard drive.

- Fixed the default table for floppy only systems (F&H MDOS) that that allows DSK9 be be a >1600 CRU 16 bit HRD card.

- Removed code that interfered with Mode 40/80 usage when used with composite monitors versus monochrome monitors.

- Updated MODE/Printer problem that was caused with a fix to 1.21 that was not compatible with quite a bit of software (sorry folks for the "bug"). TPA should now print properly.

- Forgot to add some notes to version 1.21. MDOS now has a variable reserved FDR capability. Now, 720K disks can store all filenames on the initial 128 sectors, while 90K disks will maintain the TI original number of reserved sectors. Intermediate size disks have varying reserved sector count.

- Use of Hoddie Eprom with Horizon RAMdisk is not recommended as supporting various 8/16 bit Horizons causes support problems. Replace Hoddie Eprom with original Horizon 8K Ram chip and use Jim Schroeder's Format utility.

- New command for MDOS 1.21 was not discussed in UPDATE docs. The Command is VERIFY ON or VERIFY OFF that can be issued from batch file or MDOS Command line. Default is VERIFY ON which is how MDOS was written. Using VERIFY OFF, MDOS will not verify disk writes (TI-99/4A style) and will allow speedier floppy access.

PROBLEM WITH v1.23

- Have recognized problem with 5 MB removable syquest hard drives with MDOS >1.14. Fixes have not been found yet but are believed to be tied to multiple hard drive sector reads or errors reading sectors >40 to >7F reported as protected.

A temporary solution has been found and requires sector editing on the hard drive. Basically, you must remove access of sectors >40 to >81. This will include mapping out the bit map sectors and sector 0 being modified to mark the new location of the FDR.

v1.21

- Fixes to text screen mode 26.5 * 80. Not all tables were "full" of data to handle all conditions.

- Jim Schroeder fixed Horizon RAMdisk support for 8/16 bit to use at CRU >1400 and CRU >1600. Drives DSK6, DSK7, DSK8, DSK9 set for these options as DEFAULT.

NOTE: REMAPS now discussed in REMAP_TXT 10.31.94 tat

NOTE: SETDISK now discussed in SETDSK_TXT 10.31.94 tat

NOTE: VIDEO ON/OFF removed version 1.50

- Added a new high speed video command to the CLI. Turns the Geneve video wait states on/off. Default is left to wait states enabled.

Usage:

`VIDEO ON`
turns video wait to high speed (0 wait states)

`VIDEO OFF`
turns video wait to normal speed (1 wait state)

`VIDEO`
displays current CLI setting.

Note: Usage of GPL will reset this value and will require resetting to VIDEO ON, despite the speed it displays as GPL is "external" to MDOS.

Also, if using GENMOD and 0 wait MEMEX, some applications will run too fast and will require setting VIDEO OFF to not lose video display data as the 9938 Video Processor can not keep up with our fast CPU.

- Revised XOP >11 by correcting coding errors, clearing the VDP EQ bit in R#46, optimizing low level operations.

- Revised memory management to be once again compatible with the Myarc 512K card and allow normal memory acquisition.

- Initial mod in V1.20 was incorrect for the Myarc 512K card. Due to no way of positively identifying memory speed since switches can be manipulated, all memory cards with >1.0 MB will have pages >40 to >7F, and >C0 to >FF be 0 wait state (original Charlton concept).

- solves HyperCopy timing problems and other "random" errors seen by people using the Myarc 512K card.

v1.20

- Applied KSCAN mode 8 fix to KEY1S
- Revised VID XOP'S >12 to >15.
- Modified XOP >35 to do range check on VDP register #.
- Modified Memory management to properly identify memory >1.0MB to be 0 wait state

VERSION 5.00

released July 13, 1996

THOUGHTS AND CONCERNS

This new MDOS contains some long-awaited fixes as well as a few needed additions. Besides SCSI devices, the new MDOS is capable of supporting the PFM and Rave devices, the Psystem, and more.

Some of you may wonder, "why is MDOS 120K and why does it use so much memory?" It's true - MDOS does use a good portion of the Geneve's memory. Mike's SCSI code increased MDOS' need for memory by 16K. 8K for code, 8K for buffers. In addition, MDOS 2.50S, 4.00S and now 5.00 ***REQUIRE*** the additional 32K fast RAM for SCSI access.

Some of you have suggested removing some routines such as TREE or FORMAT. The fact is that removing these and/or other routines would not allow us to shrink MDOS. The bulk of the code is found in the libraries of functions which make up MDOS; device handlers, video routines, DSR routines, and more. To shrink MDOS by 8K would require a lot of code shuffling, something which could "break" MDOS/GPL and the programs they run.

So we face the memory issue once again. In the future it may be possible to have MDOS detect whether a SCSI or HFDC is in the system then use only the memory/pages needed for the cards. However, if you have one or both of these cards and you need more memory, then buying expansion RAM is the way to go. Whether you buy a Memex, Myarc 512, Rave, or 384K upgrade you will eliminate your memory shortage.

Is that a realistic option - buying memory? I would think so. But to show you that I'm not pushing you to buy more memory, here's the system I use to PROGRAM and run GPL/EXEC/Advanced BASIC on!

- Geneve w/32K fast RAM
- HFDC, 40MB and 20MB hard drives
- Myarc Floppy controller
- SCSI controller
- Horizon RAMdisk (800K)

See - I'm using a very basic system. In fact, I've NOT upgraded so that I can continue to run/develop programs that will work on a basic system. Trust me...additional memory is nice, but MDOS 5.00 will work equally well without it. Anyway, enough editorializing...let's dig into the good stuff!

NEW/UPDATED COMMANDS/ROUTINES:

- SCSI Support added/refined, will work with SCSI hard drives and ZIP/EZ drives!
- SHIFT-SHIFT-CONTROL works with all versions of PFM.
- Fixed bug present in MDOS 2.50, caused MDOS to lock during powerup.
- PFM devices not present in 2.50S added to 5.00
- RAM count for pages >C0-EF modified to count them as FAST RAM.
- Re-added Jeff Whites keyboard filter to the MDOS keyscan routine. Keyboard processing now operates the same as GPL and EXEC 2.11

- REMAP

Typed in alone, remap now displays the current mappings for each drive number. To save space, only the drive number and map letter (as shown in REMAP_TXT) are displayed.

- SETDSK

Typed in alone, SETDSK now displays the current parameters for each drive.

- PSYS (new)

Use this command for Psystem emulation. Jerry Coffey has tested this with the psystem development system and is putting together a package for Geneve users. Syntax is as follows:

```
PSYS
```
displays current setting

```
PSYS ON
```
Enable psystem emulation

```
PSYS OFF
```
Disable psystem emulation (DEFAULT)

FIXED COMMANDS / ROUTINES:

- GENMOD SUPPORT

Jeff White fixed SCSI Genmod support, MDOS 3.00 bravo is born!

- COPY CON

Creation time stamp routine corrected, no longer uses the last time/date created in the buffer but goes out and checks the current time.

- SAVE PROGRAM (under Advanced BASIC / Extended BASIC)

Fixed problem with SAVing files. Before, saving a file over an existing file would blast the create date. Now the creation date is preserved, even if the file being saved is a different type than that on the disk (ie, saving a XB IV254 file over the existing PROGRAM file).

- CHKDSK /F

Jeff White fixed the display routine for hard drives with capacities in excess of 99MB.

Now writes back the corrected bitmap if bad sectors are found on the FLOPPY being verified. If the disk is write-protected, MDOS will display an error message.

- DISKCOMP

Bad sectors on source/target disks caused this routine to increment the TOTAL number of sectors by one for each bad sector, thereby causing the routine to loop indefinitely (verified w/2.21)

Now both source and target disks are checked for errors. Should an error occur, it is reported, and that sector skipped.

- DISKCOPY

Bad sectors caused the DISKCOPY command to fail similarly to the way DISKCOMP failed (verified w/2.21)

The bugs have been removed, and this command will copy disks properly. Bad sectors are skipped on the source. The bad sectors not read on the source are written to the target using whatever is in memory at the time of the copy.

Errors on the TARGET diskette will cause the DISKCOPY to skip writing of that particular sector. This means that the data did not copy correctly, and it is time to (a)reformat the target disk or (b) throw the target disk away and get a new disk!

- CALL DIR(), EXTENDED BASIC w/OTHER RAMIFICATIONS!

After TWO days of searching I finally tracked down a long-present DSR bug. This bug caused the CALL DIR() to lock up whenever it was used with an emulate file or a HFDC floppy.

In the process of debugging, I learned that the CALL DIR code is located in two areas, one for all floppy support and the other for HFDC. The HFDC CALL DIR code is not used but could be added at a later date. Directories are not displayed, but that too could be added later.

Removing this bug should stabilize other HFDC Floppy operations.

- CHKDSK *

Checks for fractured files on FLOPPIES only. Because code for this operation is floppy-specific, hard drives are not supported.

- RAVE RAMDISK SUPPORT

1. A Rave RAMDISK using the default MDOS configuration will give you 32K more memory showing up at pages >B9, >BD, >BE, and >BF.

2. I could not automatically check for the Rave - there was no room for the routine. So, to use the Rave as a RAMDISK, you have to do some sector editing.

Search MDOS 5.00 for HEX String: >9640

To activate the Rave RAMdisk, replace the string with >0000.

3. A formatting routine is available for the Rave but is not included here.

MDOS VERSION 6.00

Released May 16, 1998

Contact Information:
Tim Tesch
1856 Dixie Road
Port Washington, WI 53074 ttesch@juno.com <--> ttesch@execpc.com

THOUGHTS AND CONCERNS

MDOS 6.00 does not contain a lot of thrills and frills, but does contain some features some may find interesting. Most of my time has been spent on the road, visiting places throughout Wisconsin, Illinois, Iowa, and Minnesota. Perhaps this summer I'll get a reprieve to work on my favorite, though yet unfinished program, PORT. Enough editorializing :) ... on with the show!

One major undertaking was the removal of the "TI ON/OFF" command, implemented by Beery Miller some time ago. Beery implemented this command so that people could use the HFDC or SCSI at CRU >1200, and access it as "WDSx".

Unfortunately, these routines were one source of the trouble most folks have with the HFDC, and as such, were removed. I was very careful not to disrupt any existing HFDC/SCSI code, and learned a few things along the way.

Anyone using the "WDSx" feature will need to continue using the older versions of MDOS or change the manner in which they access their devices. If all goes well, the remaining issues within the DSR will be resolved with the next release.

GPL was not modified for MDOS 6.00 as there was really nothing to add or fix. However, I have modified the version and included it with MDOS 6.00 to keep things straight.

Finally, anyone wanting to use MDOS 6.00 ***MUST*** have the extra 32K RAM on their Geneve. I started modifying MDOS to remove this restriction, but since Mike hard-coded multiple locations, the task was not one I wanted to deal with. Therefore, if anyone would like me to modify their Geneve, or send them a stack of 64K memory, the cost is $10.00 plus shipping/handling. Turn-around is one business day.

NEW/UPDATED COMMANDS/ROUTINES:

-SCSI REMAP COMMAND

Some SCSI devices won't respond to SCSI ID 0, 1, or 2. To fix this problem, I've added a new command (CYA compatible) which remaps a specific ID to SCS1, SCS2, or SCS3.

Command: SCSMAP <drive><device>

Examples:

`SCSMAP 13`
Maps SCSI ID 3 to SCS1.

`SCSMAP 26`
Maps SCSI ID 6 to SCS2.

Note that remapping a SCSx drive to the SCSI controller ID could cause a lockup!

- VIDEO Command again available

After being removed some time ago, I've placed the VIDEO command back into MDOS. This command turns the video wait states on/off. While it reports the current state of the command, please note that it ***may*** report the status incorrectly. This command was added as a convenience.

`VIDEO FAST`
turns off wait states

`VIDEO SLOW`
turns on wait states

- MDOS Character set expanded

The MDOS internal character set has been expanded to 255 characters. This means that characters 128-255 are now defined and available using the following command:

```
IBMGRF <on/off>
```

IBMGRF OFF - "Normal" characters 0-127 (default) defined. Characters 128-255 are not redefined.

IBMGRF ON - ALL characters 0-255 defined. This causes at least one program, Clint Pulley's Directory Manager, to look "wrong". Clint defines his characters before asking MDOS to define the remaining chars, hence the screwup. I suggest creating a batch file for DM which turns IBMGRF off/on before/after using DM.

Using this command eliminates the need for "IBMGRAPHICS", a program I released a few years ago, and give you the added benefit of retaining those characters at all times. Additionally, CYA will imbed your favorite character set within MDOS! Thanks goes out to TONY KNERR for his idea and his original program, CHARAPATCH!

- RAVE RAMDISK

CYA will now modify MDOS 5.00 and 6.00 with respect to the Rave RAMdisk. No more sector editing required.

- SETDSK / REMAP / HARD <on/off>

Modified text output shown during an error. Changed display in HARD command; now shows current status.

-PFM512 Device Support

Support for the 512K PFM FLASH chip was added.

- MIRROR COMMAND

MDOS MIRROR command has been REMOVED. Instead, please use the accompanying file SAVEIMAGE. This program will let you save sectors 0-33 (CFORM compatible) from any HFDC or SCSI device to any other device. LOADIMAGE, used to restore the sectors, is not released here, as I do not feel comfortable giving that out with MDOS. If you want a copy, feel free to ask. Docs are included with SAVEIMAGE.

FIXED COMMANDS / ROUTINES

- YEAR 2000

(a) The DIRectory routine now displays the full year.

(b) The clock code was modified to fix a Year 2000 bug. Dates through 2050 were checked, including dates falling on a leap year, and were found accurate. I am happy to say MDOS 6.00 is Year 2000 "compliant," though there may be programs used on the Geneve which are not.

(c) Researched possibility of changing clock code to give user the full year. Decided not to make change, as it would likely break multiple applications which rely upon the 8-character output string.

- RAVE RAMDISK

Modified low-level sector code. Stabilized the access routine, which at times caused corruption in the first 32K.

- HFDC / Winchester Code:

As discussed above, removed some offending code (Winchester) from MDOS. Also removed TI ON/OFF capability, which previously used to access the HFDC/SCSI at CRU >1200 as "WDSx". This has cleared up some conflicts between the SCSI and the HFDC.

That's all this time, folks! If you have any requests or suggestions, please send them my way! Use the addresses listed at the top of this document. Requests for Source code to MDOS may also be sent to this address.