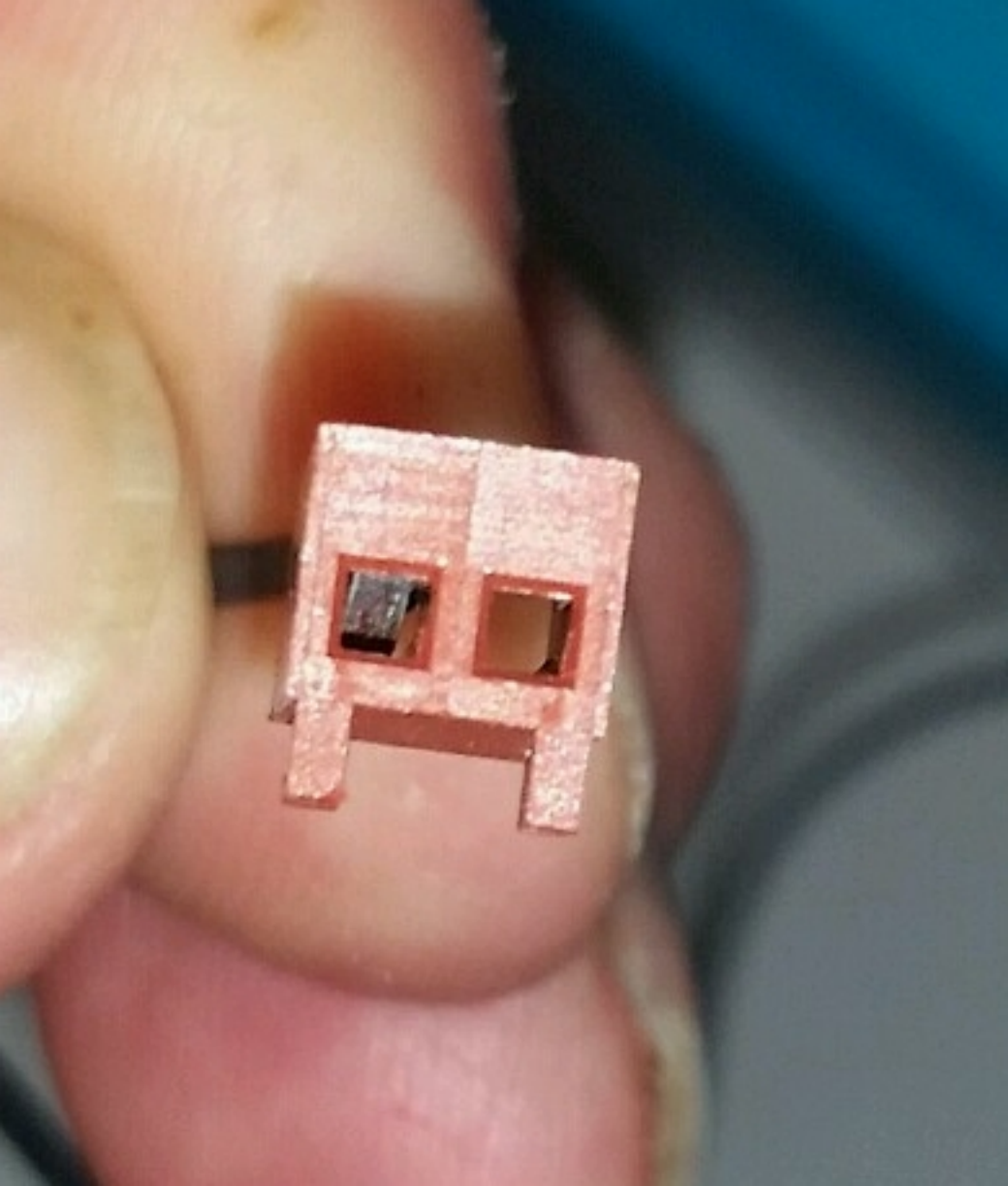
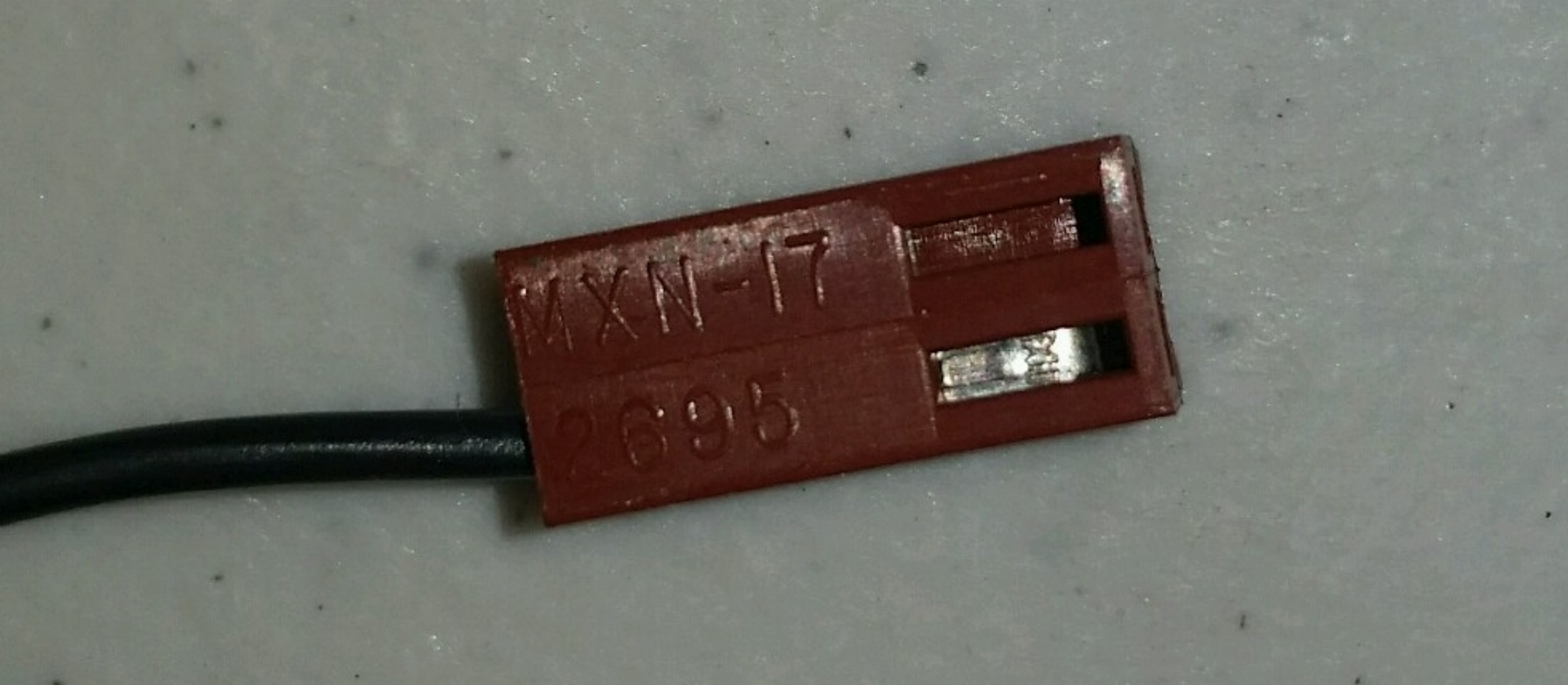

CMD
CREATIVE MICRO DESIGNS, Inc.




MXN-17
2695

CD74HCT158E
RCA H 610

2500 mAh
NICKEL-HYDRURE MÉTAL
NH15-AA
1.2V NiMH HR6

Made in Japan for: / Fabriqué au Japon pour :
Energizer Holdings, Inc., St. Louis, MO 63141

Energizer
RECHARGEABLE
2500 mAh
NICKEL-METAL HYDRIDE
AA

Made in Japan for: / Fabriqué au Japon pour :
Energizer Holdings, Inc., St. Louis, MO 63141

Energizer
2500 mAh
NICKEL-METAL HYDRIDE
AA

TRANS

PLUG-IN CLASS 2
TRANSFORMER

Power

MODEL NO.: AD-0950
INPUT: AC 120V 60Hz 9W
OUTPUT: DC 9V 500mA

CAUTION:
INDOOR USE
ONLY

UL®
LISTED
50J5
E87297
OEM

LR57562
1191

MADE IN TAIWAN R.O.C.
BY

IMPORTANT!!!

ADDITIONAL INSTALLATION INSTRUCTIONS

Please Read Before Installing RamDrive

Due to unexpected wide variances in computer timing, some changes have been made to RamDrive to assure proper timing with all computers. It is necessary for some systems to connect a clip within the computer to assure proper operation with RamDrive. The majority of timing problems arise in older C-64 and SX-64 systems.

The clip installation procedure is not required but is optional for RamDrive installations which pass the HARDWARE TEST program supplied. The clip installation procedure is required for systems that fail the HARDWARE TEST program. To dertermine if a system requires this modification run the HARDWARE TEST. The HARDWARE TEST program should be run again after any hardware modifications are made to the computer system to which a particular RamDrive is attached.

There is no way to determine how long the test should run on a particular system. A run time of two hours or more will provide a reasonable test. If you have doubts about your system's hardware compatibility with RamDrive, then intall the clip.

The following procedures detail the installation of a jumper clip in your computer. This procedure requires no soldering, and only should requires afew minutes to complete. The procedure will require a Philips Screwdriver. **Note:** Some computers may need a TORX (satr head) size T10 driver. TORX drives are common and may be found at Sears and other hardware and automotive supply stores. Often a 5/64 inch allen will serve in place of a T10 TORX driver.

Additonal Clips are available from CMD for $3.00 each plus shipping.

WARNING

Always protect your equipment by discharging any static electricity from yourself. You maydo this by touching any grounded metal surface. For complete protection, use static protection devices, such as anti-static wrist wraps.

If you have any doubt about your ability to accomplish this procedure safely, take your computer to a qualified technician for installation. Neither PPI or CMD can accept responsibility for any damages caused by the user in performing any installation.

RAMDRIVE C-64 Installation

C-64 Owners Note: The C-64C model is slightly different in design and layout than the earlier C-64 computers and requires special steps during disassembly and assembly. In the procedure that follows, the additional steps required for the C64-C are enclosed within a box.

Step 1. Make sure that the computer power switch **OFF** and that any peripherals – (disk drives, etc.) are turned off.

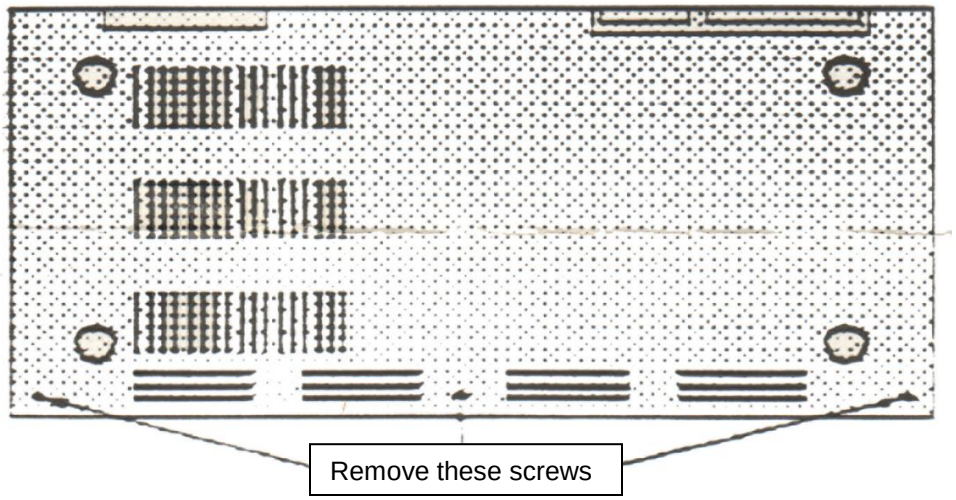
Step 2. Unplug the computer's power supply cord from the wall outlet, outlet strip, etc.

Step 3. Unplug all cables from the computer (power, joystick, serial calble, monitor cable, etc).

Step 4. Remove any devices pluged into the Cartridge, Cassette or User ports.

Step 5. Turn the computer u[side down on a suitable work surface. Remove the three screws holding top and the bottom halves of the case together, See figure below for location of these screws.

C64-C Note: The C-64C case differs slightly from the one shown below. It does, however have the same the three screws holding the top and bottom halves of the case together.



Step 6. Turn the computer back over top up. Separate the upper and lower case halves at the seam along the front of the computer and carefully lift the front edge of the upper half of the case assembly. Continue to lift until you gain access to the inside of the computer.

Step 7. Unplug the keyboard cable from its connector on the main circuit board. The cable me be a snug fit and a little difficult to remove. If so work it off carefully by alternately lifting each end of the connector.

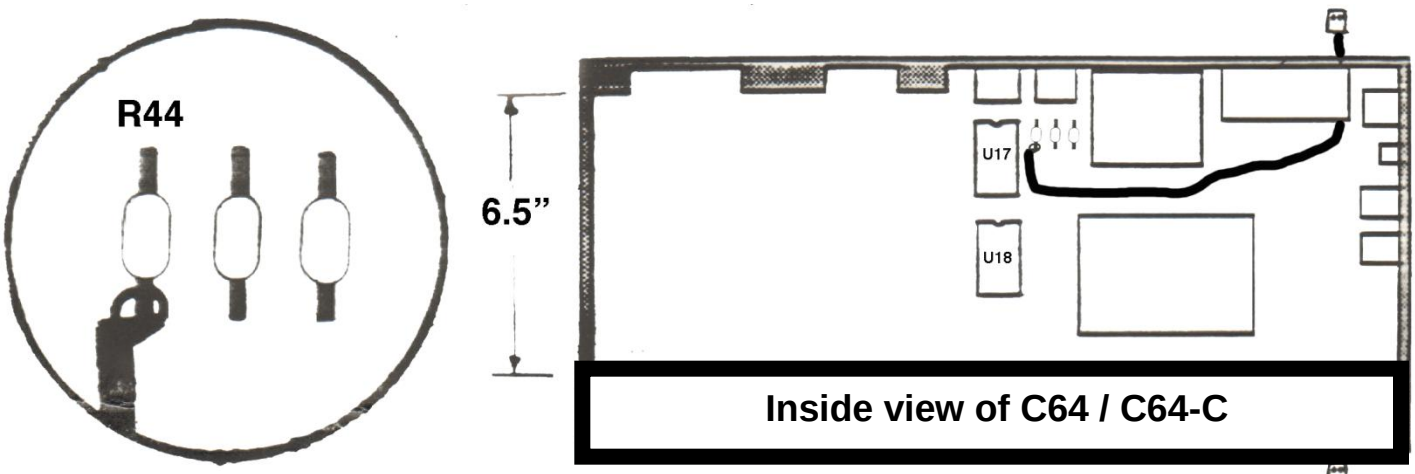
C64-C Note: Detach the keyboard from the lower case assembly by removing it retaining screws and sliding it out from under the plastic tabs at the front of the case. Place the keyboard carefully to one side.

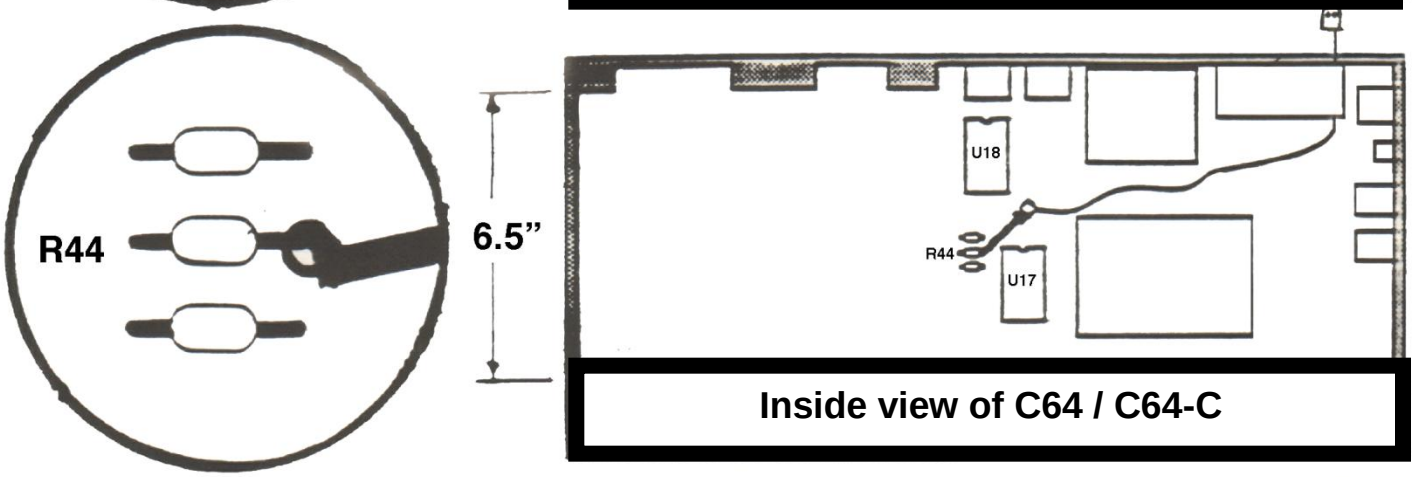
Step 8. Remove the power indicator light cable from its connector on the right side of the circuit board and then remove the upper half of the case assembly. Be careful of the plastic tabs at the rear of the upper half of the case assembly which slip into remaining slots in the lower half of the case assembly.

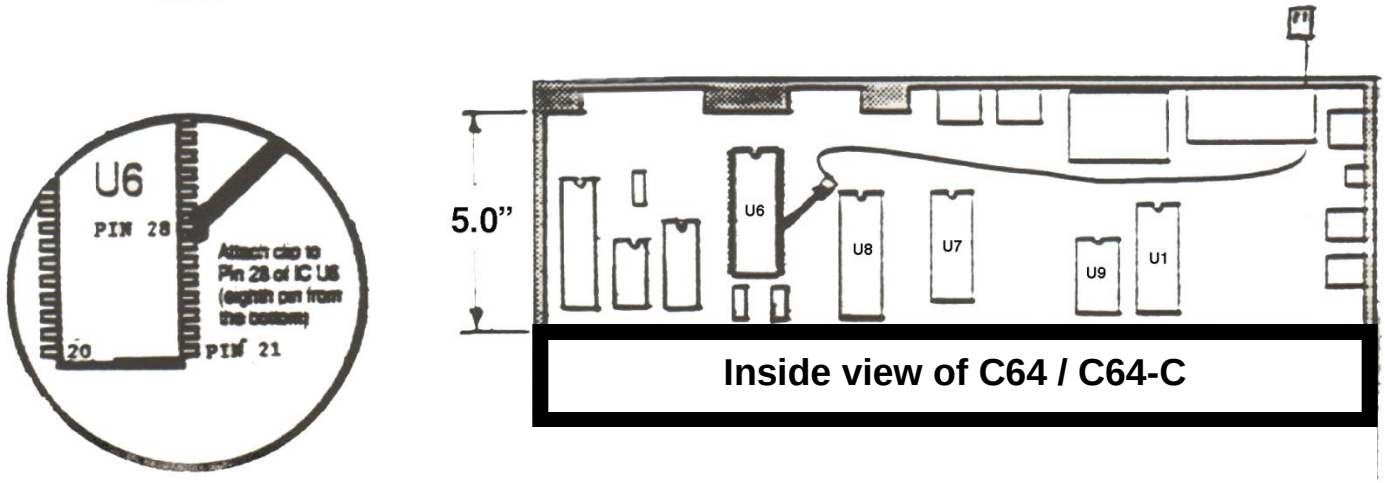
Step 9. Some 64s have a cardboard/foil or metal shield covering most of the circuit board components. If your computer has one of these shields, remove it at this time. The shield is held in by self-taping screws.

Step 10. <u>For 6.5 inches wide circuit boards (C64 and C64-C):</u> Locate resister R44 in one of the two areas shown. The exact location of R44 may vary slightly from the shown diagrams. Attach the clip to the resister as shown. Make sure that the metal portion of the clip does not touch any other pin or any other portion of another component.

For 5.) inch wide circuit boards (C64-C): Locate the pin 28 of IC labeled U6 as shown in the Diagram. Attach the clip to pin 28 as shown in the expanded view. Make sure that the metal portion of the clip does not touch any other pin or any other portion of another component







Step 11 Carefully feed the other end of the clip assembly (the end wit the xmaller connector on it) through the opening in the upper right corner of the shield located over the Cartidge Port. If you wish, you may secue the clip assembly wire near the clip with a piece of electrical Tape, by taping in to the circuit board a few inches away from the clip.

Step 12. If your computerhas a cardboard/foil or metal shield which covers the circuit board, replace it at this time.

Step 13. Take the upper half of the case assembly and rest it ina tilted position on the lower case Assembly. Make sure the tabs at the rear of the upper half of the case assembly fit properly Into their notcher in the lower half.

Step 14. Reconnect the power-on indicator light cable on to the circuit board. Make sure it mounts Properly over the pns protruding from the cicuit board.. Note: Cable orientation (position of The red and black wires)is not critical.

C64-C Note: Reinstall the keyboard in the lower assembly. Make sure the the front edge of the keyboard seats properly under the plastic tabs at the front of the lower case assembly, and then replace the keyboard mounting screw.

Spet 15. Reconnect the keyboard to its connector on the circuit board. The cable connector is “keyed” Make sure that the “key” of the connector (the positon with no hole) lines up properly with the missing pin on the circuit board holder.

Step 16. Tilt the upper half of the case assembly back down into place on the lower half. Turn your computer back over (uoside – down), and replace the three screws removed in step 5.

Step 17. Place the computer back onto an upright position, Replace power supply and other cables Which are normally attached to your computer. Remebering to plug the power supply back Into a working outlet.

Step 18. Perform RamDrive Clip Connector Installation as described on last page of this document.

RAMDRIVE SX-64 Installation

Step 1. Make sure the SX-64 power switch is **OFF**. Also make sure that any peripherals (disk drives, Printer, etc) are attached to the SX-64 are also switched **OFF.**

Step 2. Unplug the SX-64 power supply cord from its AC outlet.

Step 3. Unplug all cables from the SX-64, including (if attached), AC power supply, cable, keyboard cable, joystick or mouse cables(s), video cable, serial bus cable.

Step 4 If a disk is present in the SX-64 disk drive remove the disk.

Step 5. Remove all cartridges or interfaces from the user port and from cartridge port.

Step 6. Remove the eight screws from the rear of the SX-64.

Step 7. Remove both side rails from the SX-64 by sliding them toward the rear of the computer.

Step 8. Remove the top cover retaining screws from each side of the SX-64 top cover.

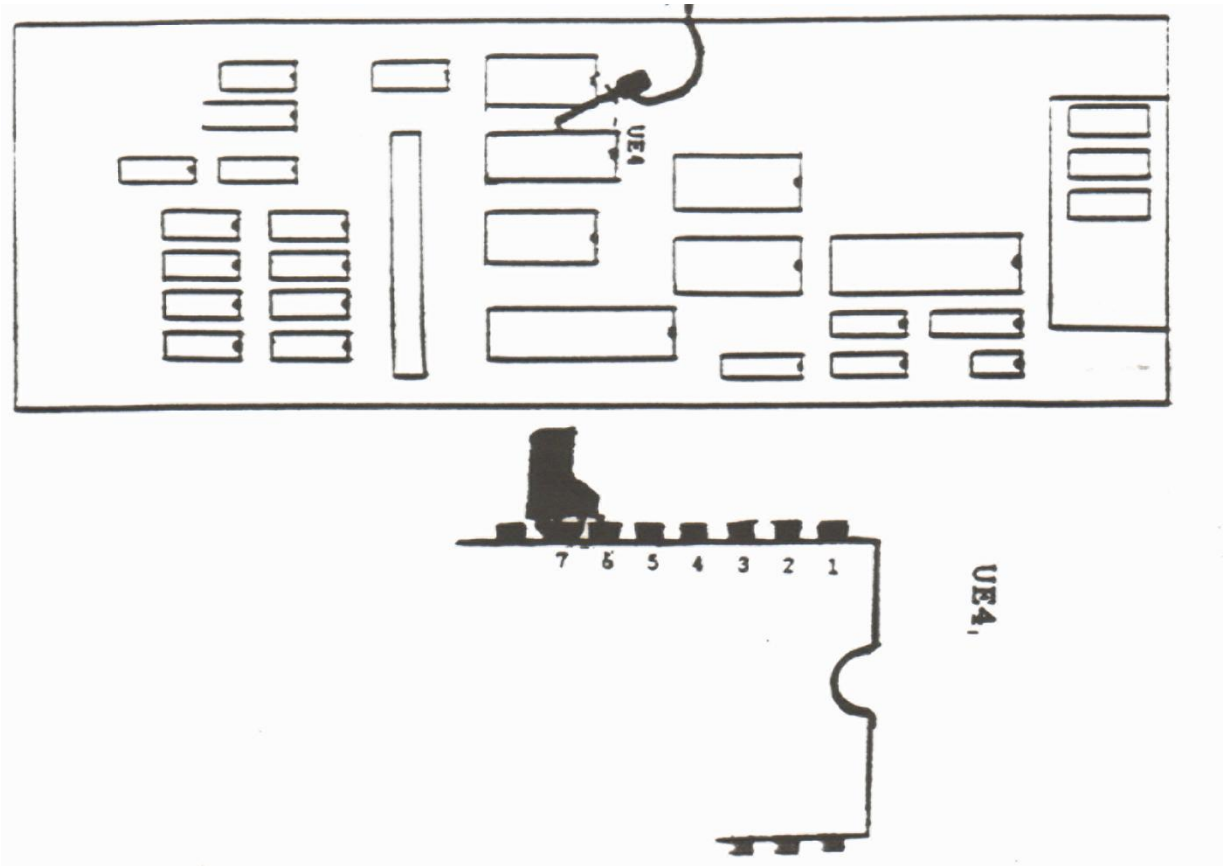
Step 9. Remove the SX-64 top cover.

Step 10. Locate the SX-64 CPU circuit board. This is the board which runs parallel to t he side of the Computer and is adjacent to the disk drive mechanism and cartridge port.

Step 11. Locate the integrated circuit labeled UE4. Attached the clip to pin 7 of UE4 as shown in the diagram below. Depress the cartridge port slots and run the wire through the cartridge port. Do not run the wire around the since the top cover will block its exit.

Step 12. At this time the computer may be reassembled. Replcace the top cover and retaining screws, replace the side rails and rear screws.

Step 13. Proceed with the RAMDrive Clip Connector Installation procedure at the end of this document.



RamDrive C-128 Installation

Step 1. Make sure that the C-128 power switch is **OFF** and any peripherals attached to the C-128 are also switched **OFF.**

Step 2. Unplug the C-128 AC power cord from it's outlet.

Step 3. Unplug ALL cables from the C-128.

Step 4. Remove all devices plugged into the User and the Cartridge ports.

Step 5. Turn the C-128 upside-down. Remove the six scres holding the top and bottom halves of the computer together.

Step 6. Turn the C-128 back over into its normal position. Separate the upper and lower case Assemblies at the seam along the left side of the computer. Lift the left side of the keyboard Assembly until it is tilted upward far enough for you to gain access to the insde of the computer.

Step 7. Unplug the indicator light cable from the left side of the circuit board.

Step 8. Unplug the keyboard cable from the C-128 circuit board and unfasten the keyboard ground strap. Place the keyboard off to one side. Note: the keyboard connection may be a bit snug fit. If so, work if off carefully by alternately lifting each end of the connector.

Step 9. Remove the screw securing the metal shield that covers the circuit board, Untwist the metal tabs around the perimeter of the shield. Remove the shield. Note: the shield maybe soldered at a point along the right side of the circuit board. If so, you can: 1 Unsolder the shield. 2. Breaker the solder by twisting the shield (no ned to solder back later): 3. Leave the soldered shield in place and bend it out of the way off to your right.

Step 10. Locate the PLA chip which is the intergrated circuit labeled U11. Attach the clip to pin 41 of this IC as shown in the diagram below.

Step 11. Carefully feed the other end of the clip assembly (the end with the small connector on it) through the opening in the upper right corner of the metal shield located over the Cartridge Port. If you wish, you may secure the clip assembly wire near the clip with a piece of electrical tape, by taping it to the circuit board a few inches away from the clip.

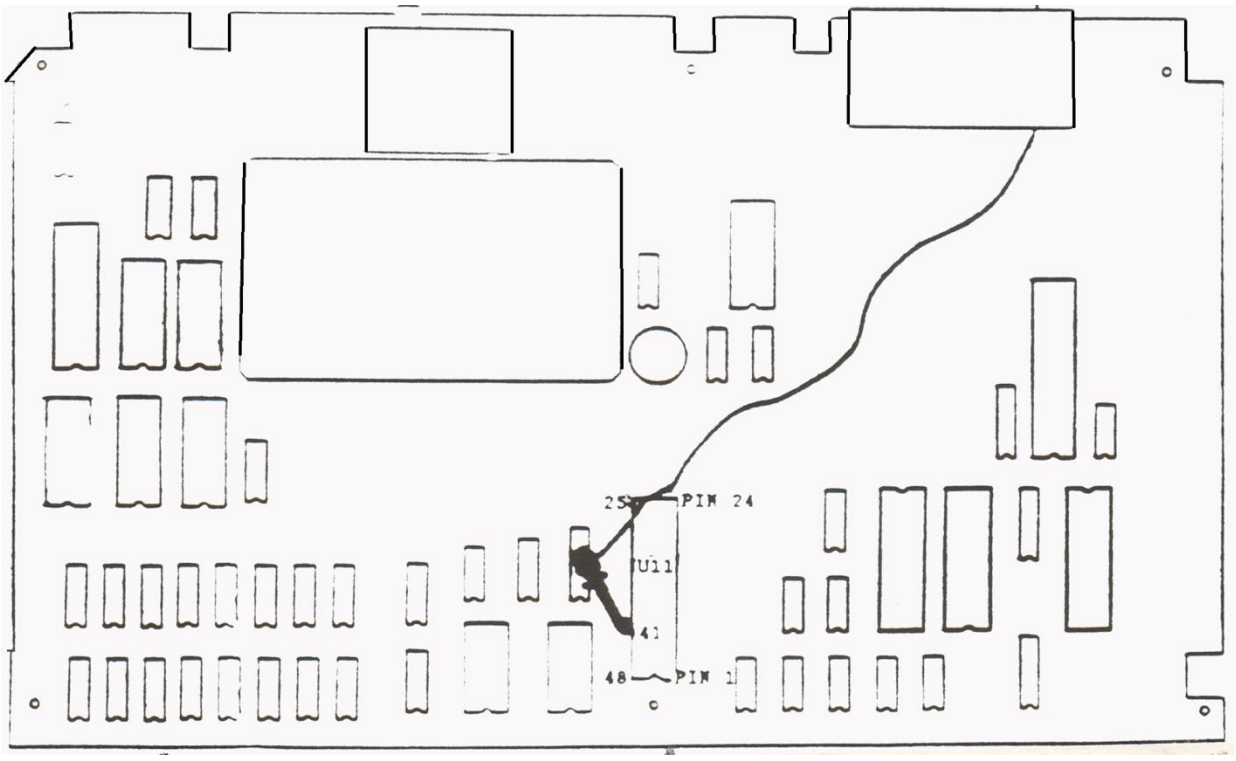
Step 12. Replace the keyboard grounding strap and reconnect the keyboard cable to circuit board.

Step 13. Tilt the keyboard assembly back towards its normal position on top of lower case assembly.

Step 14. Reconnect the power-on indicator light cable to its connector on the C-128 circuit board . Note: Cable orientation is not critical.

Step 15. Fit the upper keyboard assembly into place on top of the lower case assembly. Turn the C-128 upside down and replace the six screws which hold the upper and lower halves of the case together.

Step 16. Proceed with the RamDrive Clip Connector Installation Procedure found at the end of this document.



RAMDRIVE C-128D Installation

Step 1. Make sure the C-128D is switched **OFF** and that any peripherals attached to the C-128D are also switched **OFF**.

Step 2 Unplug the C-128D AC power cord from the outlet.

Step 3. Unplug all cables from the C-128D and remove any devices plugged into the Cartridge and User ports.

Step 4. Turn the C-128D upside down and remove the top two cover screws located on edge of the case near the front corners.

Step 5. Turn the C-128D onto its upright position and remove the three screws on the back, on the left top and right outside edges where the top cover folds over onto the back of the case.

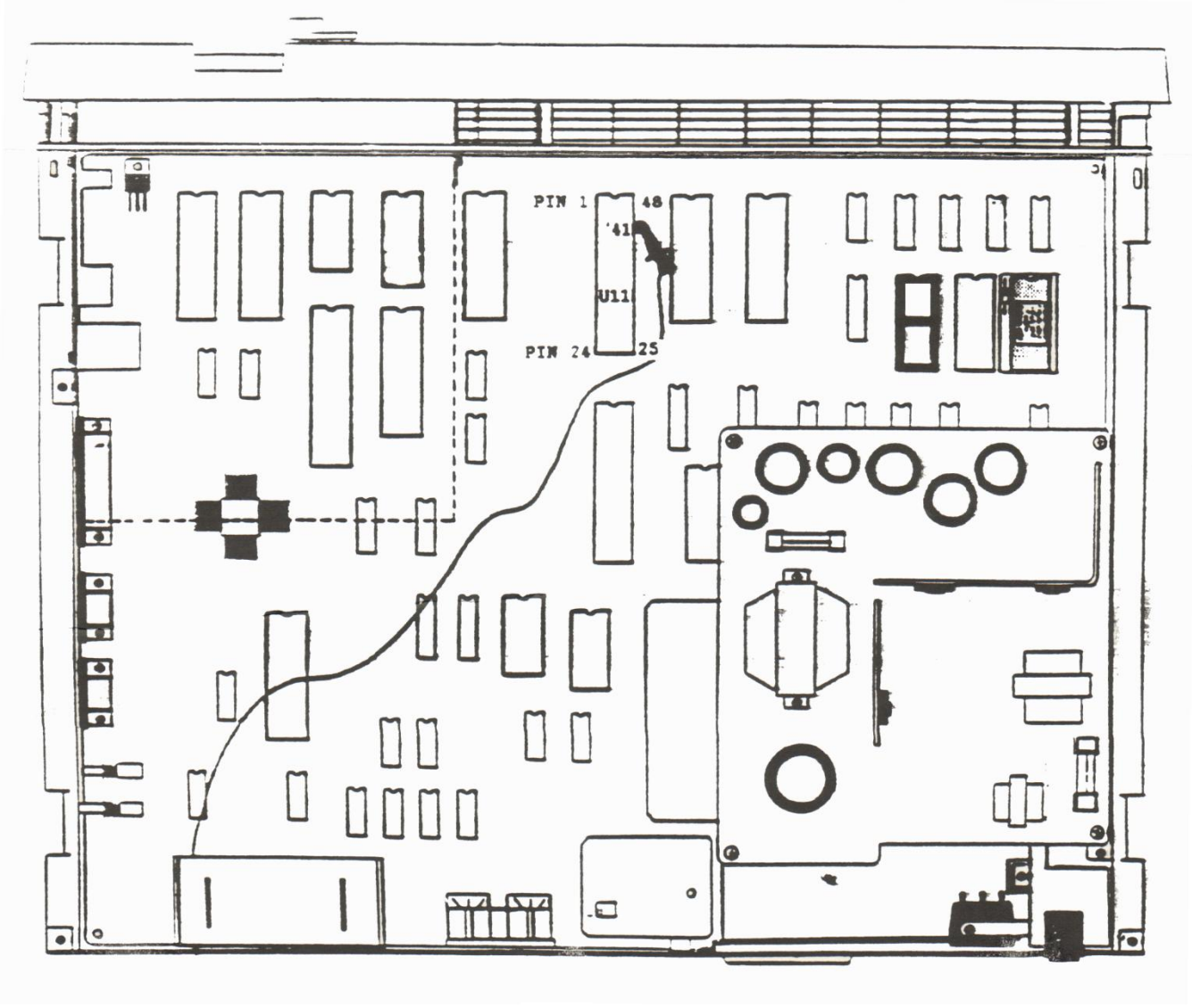
Step 6. Slide the top cover of the C-128D back about one inch towards the rear of the computer. Then Remove the top cover by lifting it straight up.

Step 7. Locate the PLA chip U11 and connect the clip to pin 41 as shown in the diagram below.

Step 8. Carefully feed the other end of the clip assembly (the end with the small connector on it) through the opening in the upper right corner of the metal shield located over the Cartridge Port. If you wish, you may secure clip assembly wire near the clip with a piece of electrical tape, by taping it to the circuit board a few inches away from the clip.

Step 9. Replace the C128D top cover. To do this, first lower the cover straight down back about one inch, so that the tabs and holes on the top cover and base assembly align. Next slide the Cover forward and into position. Replace the five mounting screws that secure the cover below and at the rear.

Step 10. Continue with the RAMDRIVE Clip Connector Installation Procedure , see last page.



RAMDRIVE Clip Connector Installation Procedure:

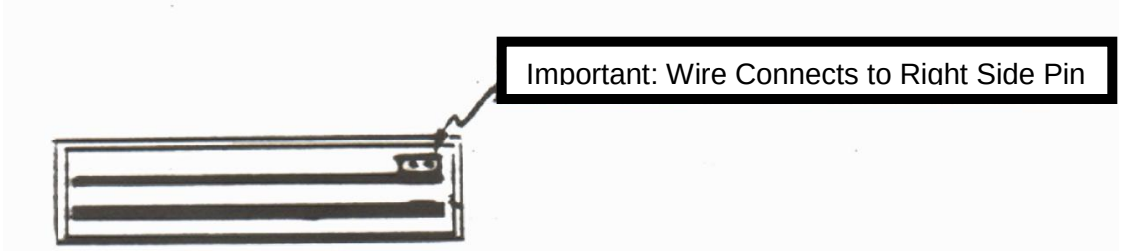
Step 1. Locate the connector end of the Clip assembly, which should be near the rear of your computer On the end of the wire coming out of the Cartridge Port opening. Hold the connector with the two tabs pointing down, and the open end pointing away from the rear of your computer.

Step 2. Remove the short jumper located on the end of the connector inside the RAMDRIVE by simply pulling it off the two posts of the connector. Save the tiny jumper in a safe place for possible later use.

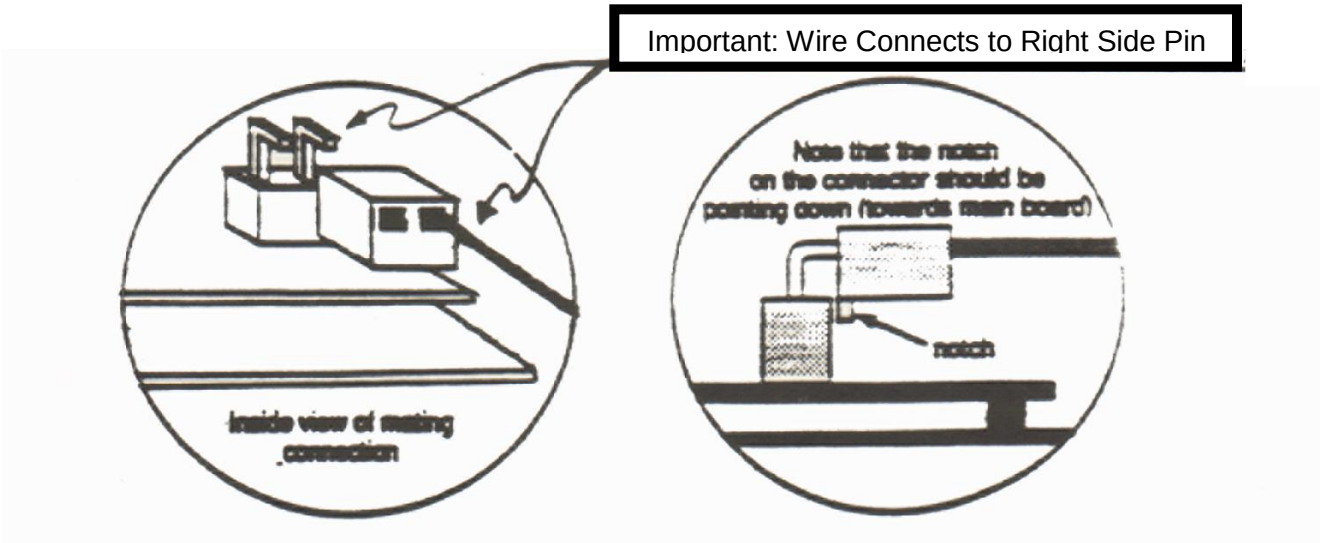
Important: Save this jumper. The shorting jumper must be replaced when using your RAMDRIVE on a computer which does not need a clip installed. Also the shorting jumper must be installed on RAMDRIVE to pass the HAREWARE TEST to determine if a clip installation is necessary before using your RAMDRIVE on other computers

Step 3. Place RAMDRIVE near the computer's Cartridge Port and hold it so that you can see the mating connector just inside the throat of the Cartidge Port mating connector. See the diagrams below to help locate this connector.

.



Step 3: Slide the Clip Assembly connector onto the mating connector inside RAMDRIVE, making sure that the pins are properly aligned.



You may now continue with the normal RAMDRIVE installation instructions in your RAMDRIVE Users Manual. Make sure that when you plug your RAMDRIVE into your computer that the wire on the Clip assembly does not interfere with the connection or become kinked or damaged.

Using GEOS with RAMLink and RAMDrive

Getting Started

This documentation will take you step by step through the procedures for getting RAMLink or RAMDrive operating under all 2.0 versions of GEOS. This includes GEOS 128, and the GEOS supplied with Berkeley's GEORAM. Unless differences exist, all versions of GEOS will be referred to as GEOS.

What you'll need...

Required Software:

- Any 2.0 version of Berkeley Softworks' GEOS (boot disk)
- RAMLink or RAMDrive GEOS Utilities disk from CMD

Required Hardware:

- Commodore 64 or 128
- 1541 or 1571 disk drive
- CMD RAMLink or PPI RAMDrive with at least 512K of RAM
- Mouse or joystick

Optional Equipment:

- Additional drives, such as Commodore's 1581 3.5" disk drive or a CMD HD Series hard drive
- A GEOS supported printer (a list of supported printers can be found in your GEOS 2.0 manual

Important: If you are using a RAMLink with less than 512K of RAM (1700 or 1764 REU only and no RAMCard memory), you cannot use the new CONFIGURE files supplied with RAMLink. Instead, switch RAMLink to DIRECT mode and use GEOS as you normally would.

About the Installation

Installation is not a complicated procedure. However, because you will be replacing files on your GEOS boot disk, we suggest you read this section over until you are comfortable with the procedures.

The installation consists of two basic parts; first it will be necessary to configure your RAM device for use with GEOS. After completing the

hardware setup, you will need to install a new CONFIGURE file. The entire procedure should take you only a few minutes.

Hardware Setup

If you have not yet installed your RAM device into your computer and checked it for proper operation, do so now. After you are certain that your RAM device is working correctly with your computer, you may proceed with the following procedures.

1. With your RAMLink or RAMDrive attached, turn on your computer. Make sure that the RAM device is enabled, and if you are using a RAMLink you should also make sure that it is in NORMAL mode.

2. Insert the disk named RAMLink GEOS Utilities or RAMDrive GEOS Utilities into your disk drive. (This is normally located on the back side of the RAMLink Utilities or RAMDrive Utilities disk.)

3. Now type the following:

   ```
   LOAD"RL/RD GEOS SETUP",8
   ```

   Press the RETURN key when you have finished. (If your disk drive is not device number 8, then substitute the correct device number.)

4. After the program has loaded, you will see a READY. prompt appear on your screen. Now type:

   ```
   RUN
   ```

   After typing this, press the RETURN key to start the program.

5. As the program begins, a warning message will be displayed. Please be aware that this program will reconfigure the partitioning of your RAM device, and all data currently stored in the device will be lost. Press the 'Y' key on your computer if you wish to continue.

6. After a few seconds, the program menu will appear. Depending on how much RAM you have available, you will see from one to six options. The following is a brief description of these options:

   (1) 256K DACC (RAM 41 ONLY) - This option will allow your RAM device to emulate a 1764 REU under GEOS. This is the only option available to RAMDrive 512K owners. All leftover RAM will be placed into a CMD Native Mode partition which may be used for other (non-GEOS) purposes.

   (2) 512K DACC (RAM 71 ONLY) - This option allows your RAM device to emulate a 1750 REU under GEOS. This option is only available if your RAM device has at least 1 Megabyte of RAM. All leftover RAM will be placed into a CMD Native Mode partition which may be used for other (non-GEOS) purposes.

(3) 64K DACC + 1581 PART. - This option creates the minimum size direct access area required for GEOS, and also creates a 1581 Emulation partition. This partition may be used both from GEOS and from the native operating modes of your computer. Some GEOS applications (such as GEOWizard) will not work properly with this configuration, though most will. This option requires at least 1 Megabyte of RAM.

(4) 128K DACC + 1581 PART. - This option creates a larger direct access area to allow the use of programs which require an extra bank of RAM expansion memory (such as GEOWizard). It also creates a 1581 Emulation partition which may be used both with GEOS and from the native operating modes of your computer. This option requires at least 1 Megabyte of RAM.

(5) 256K DACC (RAM 41) + 1581 PART. - This option creates a direct access area large enough to emulate a 1764 REU under GEOS, and also creates a 1581 Emulation partition which may be used both with GEOS and from the native operating modes of your computer. Because this option gives you the ability to have two separate RAM 'disks' under GEOS, it has some advantages. However, since GEOS is limited to three devices at any time, it can also be confusing and awkward if you use more than one floppy drive. For those using a single floppy drive, this configuration is a good choice. This option requires more than 1 Megabyte of RAM, and will not work with applications which require an extra RAM bank (such as GEOWizard).

(6) 512K DACC (RAM 71) + 1581 PART. - This option creates a direct access area large enough to emulate a 1750 REU under GEOS, and also creates a 1581 Emulation partition which may be used both with GEOS and from the native operating modes of your computer. Because this option gives you the ability to have two separate RAM 'disks' under GEOS, it has some advantages. However, since GEOS is limited to three devices at any time, it can also be confusing and awkward if you use more than one floppy drive. For those using a single floppy drive, this configuration is a good choice. This option requires at least 1.5 Megabyte of RAM, and will work with applications which require an extra RAM bank (such as GEOWizard).

Select from the options available to you one which you feel suits your needs. Press the appropriate number key to select the option.

Please note that the RAM 41 and RAM 71 areas are unique to GEOS itself, and will not be accessible outside of GEOS. Also note that most of the options will leave you with some extra RAM, which can be used to create partitions for purposes other than GEOS. If you have

sufficient RAM, you can create additional 1581 Emulation partitions for use with GEOS by using RAM-TOOLS (each 1581 Emulation partition requires 3200 blocks or 800K of RAM). A special application (RAM_MOVE) has been provided to allow you to switch between partitions from the deskTop, and it may also be used to copy files from one partition to another.

After you have made your selection, the program will configure your RAM device according to that selection. During this configuration, the program will check to make sure that your RAM device has a device number which will not conflict with GEOS operation (it cannot be set for device numbers 8 through 11). It will also set the default partition number to partition number 1. After a few seconds, the program will end by performing a reset of the computer. Hardware setup is now complete unless you wish to create additional 1581 Emulation partitions for use with GEOS by using RAM-TOOLS (see the manual provided with your RAM device for details on using the RAM-TOOLS program). You may now proceed with the installation of a new CONFIGURE file.

Software Installation Procedures

Use the following steps to install the software required to use RAMLink or RAMDrive under GEOS:

1. First, with your RAM device connected and enabled (RAMLink users make sure that you are in NORMAL mode), power up your 64 or 128 and boot GEOS as you would normally.

2. Remove the GEOS System disk from Drive A and insert the RAMLink or RAMDrive GEOS Utilities disk. Click on the disk drive icon representing Drive A to open the disk.

3. From the menu bar, select *disk*. After the disk menu unfolds, click on *validate*. At this time, the DeskTop will validate the disk. If the DeskTop reports an error, proceed no further with the installation and contact CMD for a replacement disk.

4. Assuming the DeskTop reported no errors, remove the disk from Drive A, and re-insert your GEOS boot disk. Click on the Drive A icon to open the disk.

   At this point, it is necessary to remove the CONFIGURE file that came originally with GEOS and replace it with the appropriate version found on the RAMLink or RAMDrive GEOS Utilities disk. If you do not possess a backup of this file, back it up now! Please refer to your GEOS 2.0 manual for instructions on how to back up files.

5. Place the pointer over the CONFIGURE file icon and click once. At this point the icon will become highlighted. After a short pause, click again. A copy of the icon, called a `ghost icon', will appear attached to the pointer.

6. Move the pointer to the bottom border area of the screen, between the waste basket and the printer icon. Click once. Now, the icon will disappear from the notepad, and reappear on the border.

7. Again, place the pointer over the CONFIGURE file icon and click once. As before, the icon will become highlighted; click again and move the ghost icon over to the waste basket.

8. With the CONFIGURE file icon positioned over the waste basket, click once. Your drive will whirr for a few moments, and the CONFIGURE file icon will disappear from the border. The name of the file now appears under the waste basket indicating that the file has been deleted.

9. Remove the GEOS System disk from Drive A and insert your RAMLink or RAMDrive GEOS Utilities disk. Open it by clicking on the disk icon.

   On the notepad you will find two replacement CONFIGURE files. The file you need is determined by the version of GEOS you use, either GEOS for the 64 (64 CONFIG RL) or GEOS 128 (128 CONFIG RL). From this point we will refer to these files simply as CONFIGURE.

10. Once you know which file is appropriate to your version of GEOS, place the pointer over the appropriate file's icon and click once. Click again and drag the ghosted icon to the border. Click to deposit the file icon on the border.

11. Remove the RAMLink or RAMDrive GEOS Utilities disk, and insert your boot disk. Click on the Drive A icon to open it.

12. Place the pointer over the CONFIGURE file icon (still located in the border) and click once. Drag the ghost icon to the notepad and click again.

13. The DeskTop will now request that you insert RAMLink or RAMDrive GEOS Utilities in Drive A. Do so, and click on *Okay*. After a few moments, you will be requested to insert your boot disk. You will have to swap disks a number of times before the copy operation is completed.

13. Once the procedure has been completed, the new CONFIGURE file will appear on your boot disk. Unless you wish to copy a new version of RBOOT to you boot disk (see below), you may now turn off your computer.

RBOOT

You will also notice that the RAMLink and RAMDrive GEOS Utilities disks contain the Commodore BASIC programs RBOOT and 128 RBOOT. These files take the place of the standard RBOOT files, and were written specifically for use with RAMLink and RAMDrive. If you wish to be able to reboot GEOS from RAM after exiting to BASIC, or even after your computer has been off, replace your current RBOOT with the appropriate version. You can use the same procedure given for replacing your CONFIGURE file by simply substituting the name of the file you wish to replace. For information on how to use RBOOT, see your GEOS manual.

Setting Up Your RAM Device Under GEOS

With your new CONFIGURE file copied to your boot disk, you are ready to begin using your RAMLink or RAMDrive under GEOS. Before you reboot, it is important to note the following limitations:

- Currently, only RAM 41, RAM 71 and 1581 Emulation Mode partitions are supported.
- CONFIGURE's 1581 DirShadow option is no longer supported by the 1581 driver.
- GEOS utilizes devices numbered 8, 9 or 10, and uses device number 11 temporarily for drive swapping. RAMLink and RAMDrive are not only RAM devices, but also act as 'serial bus' devices. Since GEOS does not know how to work with RAMLink or RAMDrive as serial devices, they should only be used with a device number of 12 or higher to avoid interfering with normal GEOS operation.
- GEOS expects devices to be consecutive. In other words, on a system with a single floppy drive attached as Drive A, your RAM disk should be configured as Drive B - *not* as Drive C.

With these limitations in mind, reboot GEOS with your RAM device attached and enabled (RAMLink users make sure that NORMAL mode is selected. It is not likely that when the DeskTop comes up the first time that you will see any additional drive icons associated with your RAM device. To accomplish this:

1. Double click on the CONFIGURE icon (64 CONFIG RL or 128 CONFIG RL). After a short pause, the CONFIGURE application screen will open showing three or four boxes labeled 'Drive A', 'Drive B', 'Drive C' and 'RAM expansion:'.

2. Find the first box (using A, B and C as the logical order) which does not have a drive selected. This will usually be Drive B on a system with a single floppy drive, or Drive C on a system with two floppy drives.

3. Select within that box the option you want to set according to the way you have configured your RAM device. Here is a list of likely settings according to the option selected in the RL/RD GEOS SETUP program:

   (1) 256K DACC (RAM 41 ONLY) - Select RAM 1541

   (2) 512K DACC (RAM 71 ONLY) - Select RAM 1571

   (3) 64K DACC + 1581 PART. - Select RL/RD 1581 part.

   (4) 128K DACC + 1581 PART. - Select RL/RD 1581 part.

   (5) 256K DACC (RAM 41) + 1581 PART. - Select either RAM 1541 or RL/RD 1581 part. If you have two drive slots available, you may set one for RAM 1541 and the other for RL/RD 1581 part.

   (6) 512K DACC (RAM 71) + 1581 PART. - Select either RAM 1571 or RL/RD 1581 part. If you have two drive slots available, you may set one for RAM 1571 and the other for RL/RD 1581 part.

4. We suggest selecting the RAM Reboot option shown in the 'RAM expansion:' box. The RAM Reboot option is required if you want to be able to reboot your system from RAMLink or RAMDrive using the RBOOT program. Select the DMA for "MoveData" option only if you have a Commodore REU as part of your RAM expansion system.

5. Select *save configuration* from the *file* menu.

6. Your RAM device is now installed and ready for use under GEOS. Select *quit* from the *file* menu, and begin using your RAM device.

Using RAMLink or RAMDrive with GEOS

You will find using your RAM device under GEOS transparent. In fact, your RAM device will behave exactly as the type of drive it is emulating, with one notable exception - speed. Applications and data files will load at impressive speeds. Your RAM device also provides you with new capabilities, such as being able to retain programs and data in the RAM disk and the ability to reboot from RAM - even after your computer has been turned off!

With RAM_MOVE, your RAM device offers you access to multiple 1581 partitions. Individual partitions can be dedicated to a single application and its support files. For example, one partition may alone be dedicated to geoPublish with associated applications and clip art. Another might contain a complete applications development environment, with another containing word processing or graphics. And with RAM_MOVE, you may also copy files easily between partitions. For that reason alone, a copy of RAM_MOVE and the DeskTop should be kept in each partition that you use under GEOS.

Using RAM_MOVE

RAM_MOVE is the pivot of the RAM environment for those with enough RAM to use multiple 1581 partitions. In addition to allowing easy access to other 1581 partitions, RAM_MOVE enables multiple file coping between partitions.

Starting RAM_MOVE is as simple as double clicking on its icon. After a moment the application screen will open and display a list of partitions available. (Note: RAM_MOVE will only display 1581 partitions)

Partition Switching

After the application has been started, you can easily move to another partition by simply clicking on the name of the partition in the list, and then clicking once on the *Open* button. To return to the deskTop, click once on the *Quit* button.

Copying Files Between Partitions

To copy a file or multiple files from one partition to another, use the same procedure given above in 'Partition Switching' to select the source partition. After clicking on the *Open* button, the partition list will be replaced by a list of files contained within the source partition. You will also see a number of option icons appear near the bottom of the requestor box. These options are (from left to right): De-select All, Select All, Move to Bottom of List, Move to Top of List, Scroll Down one Page, Scroll Up One Page, Scroll Up one File, Scroll Down one File.

You may select any single file by clicking once on its name. Selected files are shown in reverse print. You may de-select a selected file by clicking on its name once again. You may also select or de-select a group of files by dragging the pointer over the filenames while the button is depressed.

After you have selected all the files you wish to copy from the source partition, click once on the *Open* button. The file list will be replaced by the list of available partitions. Select from this list the target or destination partition, and click the *Open* button. The files will be copied, and when done, the program will end automatically.

Creative Micro Designs, Inc.

P.O. Box 646, E. Longmeadow, MA 01028
15 Benton Dr., E. Longmeadow, MA 01028

Phone: 413-525-0023
FAX: 413-525-0147

Subject: Complimentary GateWay

Dear RAMDrive Customer,

Thank you very much for your patience and support during this very difficult time. RAMDrive, as a product, has turned into a mammoth undertaking, much larger than we could have ever anticipated. That coupled with some of the manufacturing hurdles that normally take place in the ramping up of any product have proven to require more time in getting the product to market. RAMDrive is a product that we are very proud of and hope that it is a product will meet everyone's expectations. However, if you do have any comments or problems with the unit please utilize the enclosed problem reporting log, Q'Link or our BBS. This will enable us to attend to the hundreds of other customers still waiting for their units and will avoid costly phone calls. We have limited resources and with your help we can service everyone more efficiently.

All RAMDrive customers are entitled to a free copy of gateWay/64 or gateWay/128. It is important to note that RAMDrive is not GEOS compatible without the use of gateWay. If you have not yet received your complimentary copy, please complete the attached gateWay request form and enclose payment for $3.00 to cover shipping. This offer is only good for a period of 90 days from date of shipment. Also, please be sure to return your warranty registration card. If there are any future upgrades we will mail from the registration list.

Once again on behalf of the entire CMD staff I would like to thank all of the RAMDrive customers that have been patient and understanding. CMD is genuinely committed to the 8 bit market and with your future support and help we can continue our tradition of offer the most powerful, highest quality products for the classic line of Commodore computers.

Many Thanks,
Charles A. Christianson
Vice President

(detach and return)

Customer name: _____ RAMDrive Invoice number: _____

Address: _____ City: _____ State: ____ Zip: _____

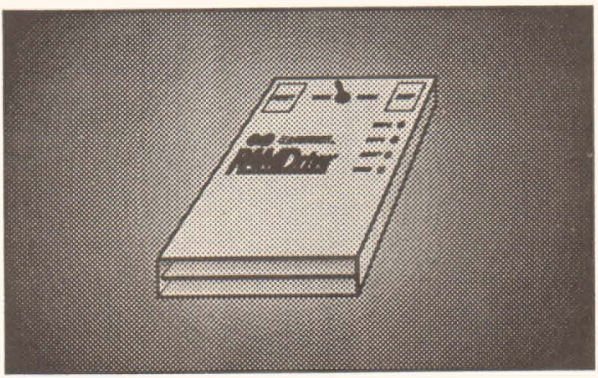
Please indicate below which complimentary copy of gateWay you would like to receive. **There is a shipping and handling fee of $3.00** and each RAMDrive customer is entitled to one version. If you would like both versions the cost of the second copy is $29.95 and can be shipped without any additional shipping charge beyond the $3.00 charge.

___ gateWay / 64 - $3.00 S/H ___ gateWay / 128 - $3.00 S/H ___ gateWay 64/128 - $29.95+$3.00

PERFORMANCE PERIPHERALS inc.

RAMDrive™

USERS MANUAL



Copyright Notice

First printing, March 1991

Table of Contents

Section 1: General Information

Section 2: Installing RAMDrive

Section 3: Special Features

Section 4: Partitions and Subdirectories

Section 5: JiffyDOS

Section 6: Using Software

Section 7: Command Reference

Appendices

Appendix A: Utilities

Section 1
General Information

Introduction

RAMDrive is a portable RAM disk for the Commodore 64 and 128 series computers. Some of the standard features include built-in JiffyDOS Kernal routines, an external power supply, and a built in rechargeable battery backup unit. Compatibility with the widest possible range of software products has been incorporated through the installable RD DOS system software.

Main Functions

The main functions provided by RAMDrive are:

- RAM EXPANDER - Internal RAM capacity allows using RAMDrive as a stand-alone REU.
- RAM DATA RETENTION - Separate power supply and internal rechargeable batteries allows RAMDrive to retain data and programs when the host computer is off and when the AC main supply off.
- JIFFYDOS - Built-in JiffyDOS Kernal routines (equivalent to the computer portion of JiffyDOS) for both the Commodore 64 and 128.

About RAM Disks

Many popular computer systems support creating and using RAM disks. Often, this is accomplished by using part of the internal RAM, such as on Amiga and IBM computers. Because of the limited amount of RAM within the Commodore 64 and 128, it is preferable to add external RAM for a RAM disk. A few companies have produced such RAM expansion units, yet most of these have been ineffective, suffering from incompatibilities with popular software. Two of the main reasons that RAM expanders have not delivered the compatibility necessary for most software are that they have lacked proper DOS emulation and have not been transparent to the system. RAMDrive overcomes these shortcomings by providing a transparent DOS interface for access to the external RAM.

Accessories: JiffyDOS Drive ROMs

The computer portions of JiffyDOS for both the 64 and 128 are contained within RAMDrive's RD DOS. By adding JiffyDOS drive ROMs to the floppy disk drives on your system, you will be able to speed up the transfer rate of the enhanced floppy disk drives by as much as 15 times. Drive ROM installation is simple and quick, and included are complete installation instructions for every drive type supported. CMD supplies ROMs for eighteen different models of Commodore or compatible serial bus floppy disk drives.

Orders and Information

If you wish to place an order or need general information about any product available from CMD, you should call Monday through Friday 10:00 am through 5:00 pm ET. If possible, call before 3:00 pm ET for fastest service. The number to call is (413) 525-0023.

You may also call this number during these hours if you have a problem with an order you placed with CMD. If you have a problem with an order which you placed with one of CMD's dealers, you should contact the dealership.

If You Need Technical Assistance

Several forms of technical assistance are available to RAMDrive owners. You may contact CMD by modem, telephone, or mail. If you experience a problem which requires immediate assistance, contact CMD at the number above. Whenever calling, have your serial number, model number, and any other important information ready. Jot down this information on a copy of the Problem Reporting Log found elsewhere in this manual.

If you have programming questions, or need non-immediate technical assistance, the best method to use is one of the two telecommunications services where support areas for CMD customers are available. The first of these is Q-Link, the most widely used service for Commodore computer users. Our support section is in the 'Commodore Information Network' (CIN) area of Q-Link. After entering this area, pick 'Hardware Support Groups'. At the next menu, selecting 'Hardware Company Support Area' will present you with a menu containing the name of our company, 'Creative Micro Designs'. Pick this choice and you will be in our area. Q-Link is on-line from 6:00 pm until 7:00 am ET (daily) and 6:00 pm Friday through 7:00 am Monday ET (weekends). We highly recommend this service to Commodore users.

CMD also maintains its own telecommunication service. This is available 24 hours a day at 300/1200/2400 baud. The communications parameters are 8 data bits, no parity, and 1 stop bit (8-N-1 or 8-None-1). Latest updates of CMD utilities are available on this system. This system also provides support files for BBS programs and general discussion areas. The telephone number for this service is (413) 525-0148.

Questions left to us on either of these services are usually answered within 48 hours. If you are unable to use one of these services to contact us about technical matters, contact us at our voice telephone number given earlier.

Bug reports or compatibility problems should be handled via mail, as they are rarely fixable via telephone, and should be accompanied by hard copy detailing your system configuration, the software involved, and the steps required to repeat the problem. Use a copy of the Problem Reporting Log found at the end of this manual for reporting bugs or compatibility problems.

We would also enjoy hearing about any successes you have in using RAMDrive with various types of software.

Acknowledgements

In 1985 PPI released the EM256K, a simple bank switched memory expansion unit. Without much software support, use of the EM256K was limited. The software which was being written to allow the EM256K to emulate a 1541 disk drive, demanded vast hardware changes and improvements. The result of this effort was the C64X9 memory expander and PPI RAMDOS. The marketing of the EM256K was delayed in 1987, do to drastic DRAM price increases. In 1990 PPI asked CMD to market the product. RAMDrive is the result of further design changes made in order for the memory expander to operate under CMD's advanced RD DOS.

Peter Fiset, founder of PPI, was the computer systems engineer on the project. Thanks also to Mark Rice, Director of the Rensselaer Polytechnic Institute Incubator Facility, for his support throughout the hatching process.

We would like to express our appreciation to the crew of CMD; a special thanks to CMD founder Mark Fellows for writing RD DOS and his aid in the hardware design of RAMDrive, to Paul Bosacki whose help with CMD's RAMLink design and GEOS compatibility code overflowed to RAMDrive, to Charles R. Christianson who handled the mechanical aspects of the case design and assisted in RAMDrive production, and to Doug Cotton for his user awareness suggestions and his work with the documentation.

Section 2
Getting Started

Before you Begin

Before you attempt to attach and begin using your RAMDrive system, you should read this manual. It may not be necessary for you to read it completely, but you should at least browse through it and become familiar with the sections which pertain to you and your intended use of the device. More importantly, before attempting to attach RAMDrive to your computer system, read this section completely, and pay special attention to any warnings which apply to you and your system.

Before attempting to attach and use your new RAMDrive, please note the following warnings:

- If, the original (stock) Kernal ROM inside your computer has been replaced with a modified Kernal ROM, then either replace the modified ROM with the original ROM, or set the modified ROM to stock mode if allowed. RAMDrive requires that a JiffyDOS replacement ROM be set to the stock mode. Failure to do so could cause undesirable operation. Some custom ROMs may make it impossible to operate your computer with RAMDrive attached, due to timing problems and code mismatches.

- Do not mate or remove RAMDrive with the computer powered up. This could cause serious electrical damage to either your computer, RAMDrive or both. Note: RAMDrive itself can and should be powered with the 9V DC power supply during connection or disconnection.

- Do not turn the computer on without power to RAMDrive. The system will stay in a pre-reset condition until 9 V DC power is applied to RAMDrive. The computers power supply will try to power up RAMDrive, which may overload the computers power supply.

Hookup

If you have not yet removed RAMDrive from its shipping carton, do so now, and remove it from any protective wrapping materials. Be sure to keep the shipping carton in case you have any problems with the system and find it necessary to return the unit to CMD for any reason.

Connections

Please follow the instructions given here when connecting your RAMDrive unit for the first time. Do not skip any sections unless they do not pertain to you (don't get ahead of us - it could cause serious problems). Whenever necessary, refer to the figures and diagrams given throughout this section for assistance in locating switches and indicator lamps.

Before proceeding, make sure that your computer is turned off.

> WARNING: You should never, under any circumstances, attempt to connect RAMDrive to your computer or disconnect RAMDrive from your computer while the computer is turned on. This can seriously damage both RAMDrive and your computer.

Next, plug RAMDrive's power supply into an AC outlet. You should use a source which will remain powered at all times if you wish to keep the batteries charging or at full charge whenever your computer is turned off. Plug the DC power jack into the side of RAMDrive. It is preferable, but not necessary, to power RAMDrive with it's 9V DC wall mount supply while connecting or disconnecting RAMDrive from a computer which is **NOT** powered.

Before plugging RAMDrive into your computer, check to see that the slide switch labeled ENABLE / DISABLE is in the position marked ENABLE. If RAMDrive is disabled the unit will not respond as a RAM disk in the following procedure.

You are now ready to attach RAMDrive to your computer. Carefully slide the cartridge port mating connector on the front of RAMDrive into the cartridge (expansion) port on your computer. This port is on the far left side of your computer when viewing your computer from the rear. Push firmly to seat RAMDrive into the cartridge port connector.

STEP 1: At this point you should have read the above, and RAMDrive should be enabled, powered with its own supply on, and connected to a computer that is **OFF**. (Computers with Kernal ROM replacements, such as a JiffyDOS ROM, must be switched to stock mode.)

STEP 2: You may now turn your computer on. Allow C128 computers to power up into C128 mode.

STEP 3: If, after a few seconds, BASIC has been initialized and is operational, proceed to step 5; otherwise, check to see if the BBG indicator is on. If the BBG indicator light is off, then immediately turn the computer off and go to the Troubleshooting Tips at the end of this section before starting over; otherwise continue to step 4.

STEP 4: Press RAMDrive's reset switch until the BBG turns off, approximately four seconds, and then return to step 3.

STEP 5: At this point the BASIC screen is displayed with the familiar flashing cursor. If the BBG light is off then go to step 6; otherwise if the BBG light is on, then RD DOS has been previously installed. If the error indicator light remains on then RD DOS has been corrupted and needs to be reinstalled; go to step 4. If the error light is off then go to step 7.

STEP 6: RD DOS may now be installed by running the RD-INSTALL program found on the RD-UTILITIES disk supplied. C128 owners should run this in the C128 mode. During the load the activity light will pulse on a few times. Once fully loaded the BBG indicator will light up, and the computer will automatically go through a software reset. You should see a BASIC screen displayed with the JiffyDOS copyright notice. Installation is now complete, and RAMDrive should be accessible as device 16.

Reloading RD DOS will not destroy valid partitions and files already present on RAMDrive.

Verifying RAMDrive Operation

STEP 7: To verify that RAMDrive is working, your first indication should be that the standard message which is normally seen on the BASIC initialization screen will be slightly different and will contain a JiffyDOS copyright message. If the message which appears does not contain the JiffyDOS copyright message, make sure that RAMDrive is enabled (check the ENABLE / DISABLE switch). You may also attempt to read the directory from RAMDrive by typing in the following:

```
LOAD"$",16
```

Press RETURN, and when you receive the READY prompt type:

```
LIST
```

Again, press RETURN and you should see a directory from RAMDrive. You can also use the JiffyDOS Control-D function until you see device number 16 appear on the screen, then use the F1 function key to view the directory (see the JiffyDOS instructions for more details on how to do this).

Partitioning & Configuration

RAMDrive performs an auto-configuration when RD DOS is installed the first time, or anytime it is reinstalled and does not recognize a valid partition already existing. Auto-configuration performs a number of tasks; it checks to see how much RAM is attached, reserves a small amount of RAM at the very top of memory, stores a configuration table, and then formats a partition.

If a Commodore 17xx series REU is used in parallel with RAMDrive, then auto-configuration will create two partitions; one for each device. See 17xx REU support section for more details.

After the installation procedures given previously, it is ready to use as is or you may use our utilities to change some of its operating parameters. RAMDrive will allow you to create a number of partitions. To take full advantage of RAMDrive for your particular needs, it may be necessary for you to delete the partitions created by RAMDrive during auto-configuration and create new partitions better suited to your needs. In order to facilitate these changes, the program `RAM-TOOLS` has been provided on the RAMDrive UTILITIES disk. The use of this program is documented in Appendix A of this manual. Please refer to this appendix when using `RAM-TOOLS`.

Configuration Parameters

The following parameters will be automatically set by auto-configuration, but may be modified by the user via the `RAM-TOOLS` program:

DEFAULT DEVICE NUMBER	The device number of RAMDrive after power on or reset. This will normally be set to 16.
DEFAULT PARTITION	The current active partition after RAMDrive is turned on or reset. This is usually set for partition 1.

Partitions

RAMDrive may be divided into sections called *partitions*. This is similar to having a number of different disks located within the same physical drive unit. Here are some general guidelines about partitions on RAMDrive:

A. There must be at least one partition present on RAMDrive in order for it to be usable as a RAM disk.

B. You may create up to 31 partitions.

C. Partition size is dependant upon the partition type. Emulation Mode partitions contain the same number of free blocks as the drive they emulate. Native mode partitions are variable in size and may be from 256 blocks to 65280 blocks in increments of 256 blocks.

Troubleshooting Tips

The following tips should assist you if you have problems with RAMDrive. Bear in mind that, as of the time of this writing, RAMDrive is still a very new device; therefore, it is difficult to foresee every possible problem.

PROBLEM: BLANK SCREEN - COMPUTER DOES NOT BOOT

POSSIBLE CAUSES AND CORRECTIVE ACTION:

The first thing to do in this instance is to turn everything off. This kind of problem is often caused by forgetting to plug something in, or by having something incorrectly attached.

- Computer or RAMDrive power supply is not plugged in or power strip turned off
  Action: Check all connections and switches
- RAMDrive is not properly connected (i.e., crooked insertion)
  Action: Check or remove and re-insert
- Corrupted RD DOS in RAMDrive
  Action: Reinstall RD DOS.

PROBLEM: COMPUTER BOOTS - RAMDrive DOES NOT OPERATE

POSSIBLE CAUSES AND CORRECTIVE ACTION:

This most often happens when you forget to have the ENABLE / DISABLE switch in the ENABLE position. Another possibility is a corrupted RD DOS in RAMDrive.

- RAMDrive power supply is not plugged in or power strip is turned off
  Action: Check all power connections and switches
- ENABLE / DISABLE switch is in DISABLE position
  Action: Check switches and correct as needed
- BBG light off.
  Action: Check BBG light. Reinstall RD DOS.
- RD DOS checksum error
  Action: Check error light. Reinstall RD DOS.

PROBLEM: INTERMITTENT OR QUIRKY OPERATION

This covers a lot of possible problems. Bad RAM, timing problems, bad power supply or power source or other hardware problems can all be causes. Software incompatibility can also cause unusual side effects, so be aware of this and test for it if possible by trying other software. Another source of unusual operation could be weak bus signals when using a 17XX series REU on an expansion bus with RAMDrive.

The best approach to take in these cases is to try other software. You should also run the RAM TEST and HARDWARE TEST programs supplied with RAMDrive to determine if problems exist with timing or with any installed RAM.

Section 3
Special Features

Default Device Number

The device number of RAMDrive is preset to 16 during auto-configuration. Unlike other peripherals used with Commodore computers, this default device number is not controlled by hardware, but is instead kept in a table located in the system partition. You may reassign RAMDrive's default device number to any device number from 8 to 29 by using the RAM-TOOLS utility located on the RAMDrive UTILITIES disk.

Default Partition

The default partition is the partition which is active after RESET has been pressed or power has been applied and is preset by auto-configuration to partition number 1. You may change the default partition at any time by using the RAM-TOOLS utility. When changing this default, make sure that you choose a partition which is actually in existence on RAMDrive.

Autofile Boot Loader

RAMDrive contains a special area in the system partition which can be used to define a number of parameters which allow a program to be automatically loaded when your computer is turned on. This file may be located on any drive attached to your system, and may be either a BASIC or a machine language program. Separate areas are maintained for 64 and 128 modes, so 128 users may have separate files for each mode. The utility called AUTOFILE EDITOR, located on the RAMDrive UTILITIES disk may be used to create the necessary configuration for programs you wish to load in this manner.

Swap Switch

One of the most powerful features of RAMDrive is its ability to swap device numbers with other disk drives on the serial bus. This function is implemented through the use of the SWAP switch on the front panel. Pressing SWAP causes RAMDrive to change its device number to device 8 or 9 and assigns RAMDrive's current device number to the drive that is normally device 8 or 9. There is one swap button that cycles through 8, 9 and then back to the default device number. The two swap lights indicate the device number that RAMDrive is swapped to. When both of these lights are off, RAMDrive's default device number is selected.

Although the swap indicator lights will change immediately upon pressing the SWAP switch, the actual swap does not occur until the serial bus is accessed. When the serial bus is accessed the device numbers are first swapped and then the serial bus access occurs respecting the resulting device numbers. Therefore, RAMDrive can be cycled through SWAP 8, SWAP 9 and its default device number without affecting other serial devices until an access is made.

As an example, assume RAMDrive is in it's default mode as device 16 and there is a 1541 drive on the serial bus that is device 8. When SWAP is pressed, RAMDrive's SWAP 8 indicator will light up. When the next serial bus access occurs, RAMDrive becomes device 8 and the 1541 is swapped to device 16, then the access occurs.

The most common use for swapping device numbers is when using software that will recognize only devices 8 and/or 9. With such programs, the SWAP function enables you to make RAMDrive device 8 or 9 before loading the software (if the program is on RAMDrive), or after loading (if the program is copy-protected and cannot be moved onto RAMDrive). In either case, having the ability to make RAMDrive device 8 or 9 guarantees that any data files used by the program can be stored on RAMDrive.

RAMDrive resets to its default device number immediately, without swapping with the other devices on the serial bus. When this occurs either reset the drive that RAMDrive had been swapped with or swap RAMDrive until both devices are accessible again. If multiple devices have been swapped, then all the units will have to be considered.

For CMD HD owners, there are a couple of things to remember about device number swapping which are not mentioned in the HD manual. For either of the SWAP switches on the HD to operate properly, you must leave the serial cable attached to the HD. SWAP commands are sent directly over the serial bus from the HD to other devices. Also note that the HD may not initiate a SWAP to RAMDrive. If you wish to swap device numbers between RAMDrive and a CMD HD, use RAMDrive's SWAP switches. Also note that you should never attempt to SWAP the HD to a device number which you are un-swapping RAMDrive from. For example, if RAMDrive is swapped in as device number 8 and you wish to make the HD device number 8 instead, you must first unswap RAMDrive by pressing SWAP on RAMDrive, perform some disk operation, and then press SWAP 8 on the HD.

Enable / Disable Switch

To use RAMDrive, this switch must be in the ENABLE position. When enabled RAMDrive electronically replaces the Kernal ROM mounted inside the computer. In the DISABLE position, RAMDrive becomes inactive, allowing the Kernal ROM within the computer to operate as it normally would without RAMDrive installed. The main reason for disabling RAMDrive would be to avoid incompatibility with some software. While this should be a rare occurrence, it is possible that you may have some software which will not operate with RAMDrive attached. If you have a JiffyDOS ROM mounted in your computer, it should be switched off when RAMDrive is enabled, but you will want to turn it back on whenever RAMDrive is disabled or disconnected.

RAMDrive should not be disabled or enabled when the activity light is on.

The position of this switch may be changed while a program is running, but if you are in BASIC, you should turn off the JiffyDOS commands by using the JiffyDOS `@Q` command before moving the switch to the DISABLE position. If you are switching from DISABLE to ENABLE, then you should also be sure to issue a SYS command to reactivate JiffyDOS on RAMDrive:

C64 or 64 mode	`SYS 58551`
C128 in 128 mode	`SYS 57651`

Reset Switch

There are two functions of this switch, depending on the length of time the reset switch is depressed. (Note: RMADrive's reset switch does not reset the internal 1571 floppy disk drive located inside 128D computers. Only the drive reset switch located on the side of the 128D can perform a drive hardware reset.)

Short Reset

A quick reset action, resets the computer and RAMDrive. If enabled RAMDrive retains control when BBG is on.

Long Reset

A long reset action, approx. four seconds, will reset the computer and RAMDrive, and will turn BBG off. Because BBG is off, control is relinquished from RAMDrive, and is given to the computer itself. To regain RAMDrive functions reinstall RD DOS or run the `BBG-TOGGLE` program supplied on the utilities disk. Note that when it is desired to make

RAMDrive temporarily inactive, it is easier to use the DISABLE switch than to reset BBG and run BBG-TOGGLE.

Only the RESET switch on RAMDrive can provide the long reset needed to turn BBG off. RESET switches external to RAMDrive have no effect on the BBG register. The BBG register may be reset by a long reset with or without power attached. BBG will always be off when powering up the computer with a RAMDrive with uncharged backup batteries.

Activity Indicator

When the activity light is active, RAMDrive is performing a DOS function. The activity light will turn on when any serial bus access occurs and during other system operations such as a reset. RAMDrive should not be disabled or enabled when the activity light is lit.

Error Indicator

The error indicator light becomes active if a full reset is not completed; i.e. a RD DOS checksum error. The regular serial bus device errors will cause the error indicator to light up. These errors can be read from the command channel. Note, unlike serial bus floppy disk drives, RAMDrive's error light does not blink.

BBG Indicator

BBG is the mnemonic for Battery Backup Good. When this light is on, it implies that RD DOS is installed and that power to the memory has been constant. If RAMDrive is enabled and BBG is on, then RAMDrive takes control of the computer system.

If RD DOS is not installed on RAMDrive then BBG should be off. If BBG is on and RD DOS has not been installed, then the system will not function. In this case, BBG has to be manually turned off by pressing the RESET switch on RAMDrive for approximately four seconds.

The BBG register is set by the RD DOS installation program. It is important that RD DOS is loaded into RAMDrive for proper operation.

BBG does not directly indicate if the batteries are charged. The 9V DC power will keep the BBG register set, while charging the batteries. It is advisable to leave the power supply to RAMDrive on until transportation is required. However, if power to RAMDrive is removed after installing RD DOS, and BBG does not come on when the computer is turned on, then this indicates that the batteries are not even partially charged.

Note: Power to the BBG indicator light is shut off when the computer is shut off. However, the backup power circuit retains the state of the BBG register when the computer is off. The light will turn on when power to the computer is on and the BBG register is set.

Three actions will cause the BBG register to be reset.

1. The BBG register will reset after the voltage to the memory drops below a set point. The data in RAMDrive may or may not remain intact below this voltage level.

2. The BBG register will reset after a manual reset of approximately four seconds on RAMDrive's reset switch. A short reset will reset the computer, and RAMDrive will retain control because the BBG register remains set.

3. The BBG register will reset under software control.

Battery Backup

The four rechargeable AA NiCad batteries inside RAMDrive protect the contents of RAM whenever a power loss occurs, or while transporting RAMDrive from one location to another. Note that it may take several hours for the battery to become fully charged. Leave it attached to RAMDrive with the power supply attached and turned on for at least 15 hours to assure yourself that a full charge has been attained. The length of time that the battery can protect your RAM from power loss is dependent upon the amount and type of RAM which is mounted in RAMDrive. A full charge should allow a backup for approximately 4 days. Note: It is highly recommended that files on RAMDrive, or any storage device, should be archived for the best protection against data loss.

There is no direct indication of the level to which the batteries are charged. Therefore always keep the RAMDrive charging by leaving the 9V DC power supply on, even when RAMDrive is not being used.

RAMDrive may be left connected to the computer during battery only backup without any significantly reduction of the backup time. RAMDrive will not power the computer or the computer's signal lines when the computer is off.

The BBG register is reset when the backup batteries near complete discharged.

Section 4
Partitions and Subdirectories

Partitions

The entire storage area of RAMDrive may be divided into a number of smaller areas called *partitions*. While this term is very common to users of larger computer systems, it may be new to many users of Commodore computers. Simply stated, the use of partitions on RAMDrive gives the appearance of using a number of separate disks, all located within the same physical device. In all, RAMDrive can handle up to 31 of these partitions. As mentioned earlier, RAMDrive automatically creates a partition during auto-configuration. Partitions may be created and deleted by using the RAM-TOOLS utility. Each partition may be defined to be one of several types. The three main types currently available for use on RAMDrive are:

Native Mode
Emulation Mode
Foreign Mode (Direct Access)

In all, RAMDrive can handle up to 31 of these partitions. As mentioned earlier, RAMDrive creates a Native Mode partition during auto-configuration. You may change your partitioning by deleting this partition and creating your own using the supplied RAM-TOOLS utility.

Current Partition

Partition 0 (zero) has a special meaning under RD DOS. It is used to indicate the current partition (the partition in which is currently active). This insures compatibility with software which issues a "0:" within filenames or disk commands. Before attempting to use most commercial software with RAMDrive, it is usually wise to select the partition you wish to use. This will assure that any further file access will occur within that partition, especially if the software does not allow you to send disk commands.

Native Mode Partitions

Native Mode partitions allow you to take full advantage of the many additional features provided by CMD's RD DOS (RAMDrive Disk Operating System) while retaining full compatibility with all standard Commodore DOS commands. This type of partition can access up to 16 Megabytes of storage space, providing the highest capacity available to Commodore DOS commands on RAMDrive. Since Native Mode partition size is variable and may also be as small as 256 blocks, it allows you to use only the amount of storage which you feel is required for a particular

partition. This mode is also the only mode which supports true subdirectories and dynamic allocation of directory space. This means you can easily organize your files and continue to add files until no free blocks remain in the partition.

Native mode partitions have 256 sectors per track, and may have from 1 to 255 total tracks. Since the size is variable, all header, BAM, and directory information must be stored on track 1. Subdirectories have a file type of DIR and have been assigned a filetype value of 6.

Emulation Mode Partitions

Emulation Mode partitions allow you to retain compatibility with software programs which require the tracks and sectors of a disk, as well as the BAM and directory, to be laid out in the same way as on a particular type of Commodore floppy disk drive. For this reason, an Emulation Mode partitions has a fixed storage capacity equal to the capacity of the disk drive that it is emulating. There are three types of Emulation Mode partitions available on RAMDrive. They are:

1541 Emulation Mode
1571 Emulation Mode
1581 Emulation Mode

As the names imply, these partitions emulate Commodore's popular 1541, 1571 and 1581 disk drives. This is accomplished by utilizing the same track and sector layout as the type of disk drive being emulated. The BAM and directory areas of these partitions are also located in the same blocks as they would be on the emulated drive. Even the internal job queue codes and locations have been duplicated. Other similarities (and a few beneficial differences) have been created within the emulation mode partitions. It is important to note that emulating these other disk drives fully in the areas of hardware and firmware mapping would have driven the cost of RAMDrive to an unreasonable level, and was therefore not attempted. All other aspects of compatibility were carefully scrutinized, and incorporated where feasible. The following paragraphs describe each of the individual emulation modes.

1541 Emulation Mode Partitions

In 1541 Emulation Mode partitions the directory and BAM are found in the same locations as they are on the Commodore 1541. All bytes within these blocks have been defined identically to their counterparts on the 1541, including the BAM bytes. This type of partition uses 683 blocks of RAM, of which 664 are free for user data or programs.

1571 Emulation Mode Partitions

1571 Emulation Mode partitions are identical to 1541 Emulation Mode partitions with only a few differences. First, 1571 partitions have twice as much storage capacity as do the 1541 partitions. This type of partition uses 1366 blocks of RAM space, of which 1328 are free for user data or programs.

There are also a number of extra bytes required for the BAM in the header block and on track 53. These have been allocated in the same fashion as they are on the standard 1571 disk drive. Also, 1571 partitions are always equivalent to the double-sided version of the 1571. If for some reason you need to emulate a single-sided 1571 disk, the 1541 partition is fully capable of doing this and should be used instead.

As with 1541 Emulation Mode partitions, it is possible to read and write to the 'super' side sector form of REL files. This method, if used, allows REL files created within 1571 Emulation Mode partitions to grow beyond the former maximum size of 726 blocks. Creating a super side sector REL file can only be done by writing a utility which creates the super side sector itself, then places pointers to any existing side sectors for that file into the super super side sector. The directory entry for the REL file would also have to be modified to point to the super side sector.

Burst commands which are normally accepted by the 1571 are not accepted by RAMDrive, and will return a syntax error. This is because the normal burst protocol requires that data be transferred directly via the serial port, which is not physically connected to RAMDrive.

1581 Emulation Mode Partitions

1581 Emulation Mode partitions provide emulation of the 1581 header, BAM, and directory information of the 1581. One area of difference between the 1581 and its RAMDrive counterpart is that RAMDrive does not utilize the track cache buffer found on the standard 1581. This was incorporated into the 1581 to increase speed - an area where RAMDrive does not suffer. Burst commands for the 1581 are not supported for the same reason given for 1571 Emulation Mode partitions. All other standard DOS commands are fully implemented, including 1581 style partitioning commands.

In order to retain full compatibility with the 1581, the DOS initialize command initiates a change in the current 1581 partition status, causing further accesses to be performed in the root directory of the 1581 partition.

Foreign Mode (Direct Access) Partitions

Foreign Mode partitions were created by CMD for use on the CMD HD for data from another type of computer. Foreign Mode partitions were included in RD DOS for an entirely different purpose - to protect areas of RAM from

use by RD DOS. This type of partition is ideal for those who wish to use the direct RAM access commands provided with RAMDrive, or if you have a Commodore 17xx series REU expander which you wish to use in parallel with RAMDrive. When you create a Foreign Mode partition on RAMDrive, you will not be able to get a directory of that partition, nor will you be able to use that partition with any other DOS commands. Only by using the Direct RAM Access routines discussed later in this manual will you be able to read from and write to this type of partition.

Native Mode Subdirectories

The following information is intended as an introduction to how Native Mode subdirectories are stored on RAMDrive. The commands used to create, remove, and move around within subdirectories can be found in the Command Reference section of this manual.

Native Mode subdirectories are similar in structure to the subdirectories used on MS-DOS types of computers. When a subdirectory is created, a DIR type file (filetype 6) is created and added to the current directory. Subdirectory names may be up to 16 characters long, just as any other filename. This "file" is initially two blocks long, and consists of a directory header block and the first directory block. These blocks are always located next to each other on the same track, and if two adjacent blocks cannot be found, no directory will be created.

The storage space available to subdirectories is the same as that available to the parent directory (the directory in which the subdirectory exists). In fact, all of the blocks within a Native Mode partition are shared between all directories within that partition. This is quite different than the method used for 1581 type subdirectories (or sub-partitions as they are referred to in this manual). This means that if there are 6000 blocks free in the partition, this number of blocks free will be indicated no matter which directory you are located in. If a 37 block file is saved in any directory within the partition, all directories within that partition will indicate 37 fewer blocks.

Subdirectories may be created in the 'root' directory (the first or main directory in that partition) or within another subdirectory. Placing a subdirectory within another subdirectory is called 'nesting'. There is no actual limit to the number of directories located in a partition, nor is there any limit to how deep subdirectories may be nested. The only limitation on creating subdirectories is the number of adjacent blocks located within the partition. There are, however, some practical limitations if you wish to be able to easily access files located in various subdirectories with a single command. This is because the input buffer of RAMDrive is only 127 characters long, therefore nesting subdirectories too deep could necessitate using more than one command string to access files within a particular directory.

Also note that when you are in a subdirectory, it is not possible to use the DOS NEW command on the partition containing that subdirectory. This was done to protect against accidental erasure due to user error. Other protective limitations have been placed on subdirectories - you cannot delete a subdirectory while there are still files located within that subdirectory, and the command for removing a subdirectory may only be issued from the parent directory of the subdirectory you wish to delete. These concepts are explained further in the Command Reference Section of this manual under the individual subdirectory commands.

Section 5
JiffyDOS

About JiffyDOS

JiffyDOS is a combined disk drive speed and DOS command enhancement system. Normally JiffyDOS is sold as a two-part system, replacing the Kernal ROM(s) in the computer and the DOS ROM in a disk drive attached to your computer. By replacing both components which control the DOS transfers within the system, it is possible to speed up all file access with a high degree of compatibility.

RAMDrive contains the computer portion of the JiffyDOS system. By integrating the JiffyDOS Kernal routines into RAMDrive in this manner, it becomes possible to use the JiffyDOS DOS wedge and command enhancements with all drives on the system. However, increasing the access speed to floppy disk drives on the serial bus requires the installation of a JiffyDOS drive ROMs inside the floppy disk drives. These drive ROMs are available from CMD, see section 1 of this manual.

A JiffyDOS manual has been included with your RAMDrive manual so that you may become familiar with the JiffyDOS commands and enhancements. Since this manual is normally applies to the regular JiffyDOS system, there are a few differences with the RAMDrive version which are documented here.

- References to the JiffyDOS switch may be assumed to be equivalent to the RAMDrive ENABLE / DISABLE switch
- The SYS commands given in the JiffyDOS manual which are used for switching to the JiffyDOS Kernal while in BASIC and for re-enabling the JiffyDOS function keys are the same for the C64 (or a C128 in 64 mode), but are different for a C128 in 128 mode. These SYS commands for RAMDrive are:

C64 or C128 in 64 mode	SYS 58551
C128 in 128 mode	SYS 57651

IMPORTANT NOTE: If you have JiffyDOS on your system already, you must switch it off on your computer when using RAMDrive with the ENABLE / DISABLE switch in the ENABLE position. If you disable RAMDrive for any reason, you may re-enable the JiffyDOS in your computer using the directions and SYS commands given in the regular JiffyDOS manual.

Section 6
Using Software

General Software Installation

There are many types of programs, but for this discussion we will separate programs into two main categories - those which are copy-protected, and those which are not. Most commercial programs, with the exception of some games, are compatible with RAMDrive in one form or another. This section of the manual will help you to discover which programs will work with RAMDrive and the steps required to get them working.

Software without copy-protection

Software which is not copy-protected can almost always be installed directly on RAMDrive. With these programs the main concern is usually which type of partition to use.

The best way to determine which partition type to use is to experiment, starting with a Native Mode partition. It is usually best to start by creating a Native Mode subdirectory to hold the files used by the program. Use RAMDrive's MD (Make Directory) command to do this and then copy all the files from the program disk to that subdirectory by using FCOPY.

To test whether the program works or not, change to the partition and directory where you have placed the program and begin running it as you would from a floppy disk. It may be necessary to set RAMDrive as device 8 or 9 by using the SWAP switch on the front panel (most software requires device number 8). If the software does not work from within a subdirectory, try using the root (or main) directory in a Native Mode partition.

Some software, even though it contains no copy-protection, will only work with a certain type of drive. Such programs will usually work in one of the Emulation Mode partitions on RAMDrive. Most often, the partition will be a 1541 or a 1571 Emulation Mode partition, though many programs may work in, or possibly require, a 1581 Emulation Mode partition.

When trying emulation mode partitions, work your way 'downward', trying a 1581 partition first, then a 1571, and finally a 1541 partition. Before testing, be sure to use the CP (Change Partition) command to select the partition in which you placed the program. If the software will still not work in any of the partitions, all is not lost, as there may be some 'hidden' data on the program disk which cannot be duplicated with FCOPY. In this case, use MCOPY to copy the entire disk to a partition of the same type.

If the program still does not work, the software is probably performing some very drive specific tasks. This is rare unless the program is some kind of disk utility. The program might also be making use of a fast loader routine written specifically for a certain type of disk drive. Fast loaders should be disabled if possible in order to get software to load directly from RAMDrive.

Multiple Disk Programs

Some programs are distributed on a number of disks. In many cases, simply copying all the files from each of the disks into a single partition using FCOPY will allow you to use this type of program on RAMDrive. Watch out for file names which are the same when copying these types of disks. These files may or may not be identical. As long as a multiple disk program uses standard files, and as long as it can determine that the proper disk is in use while searching for a file, this method will work.

If some of the information for a program is stored directly on the disk without a file name, or the disk name is checked to determine if the correct disk is being used, it will be necessary to use MCOPY to copy the disks to partitions. Also, you may only be able to use one of the disks in the set on RAMDrive if there is no provision for sending disk commands before a disk swap is to occur. The CP (Change Partition) command is the only way to move from one partition to another on RAMDrive.

Other Solutions

Some software expects to find the disk directory in a certain place on the disk. Normally this kind of software can be operated using an emulation mode partition of the required type. It may also be possible for this software to be operated in a Native Mode subdirectory which has been specially created to simulate the directory of a 1541 or 1581 disk drive. These types of subdirectories can be created in an empty Native Mode partition which has the required number of tracks by using the `1541SUB` and `1581SUB` utilities supplied with RAMDrive.

Creating an emulation subdirectory is often a good method to use to extend the amount of usable disk space with programs which must find the directory in the same area of a disk as it would on a 1541/1571 or 1581 disk drive. Be aware, however, that some programs check to see how many tracks or sectors it can access, and determine the type of drive being used by doing so. As a result, it may be difficult to determine which subdirectory type to emulate (1541/1571 or 1581). As long as you use Native Mode partitions with 40 or more tracks, 1581 emulation subdirectories will usually work with these types of programs. Some programs will always assume a 1541 or 1571 no matter how many tracks you have in the current partition.

Whenever you use a subdirectory for emulation purposes, remember to use the CD (Change Directory) command to make that directory the current one before using your software. This may be done from another partition, as long as you include the proper partition number and path in the command.

Copy-Protected Software

Normally, copy-protected software cannot be placed directly onto RAMDrive. You may, however, be able copy some of the more mildly protected programs by using MCOPY. This tends to work more often with software supplied on 1581 disks than with 1541/1571 programs.

If a program will not load from RAMDrive, you may be able file copy most of the data files to RAMDrive with FCOPY, and then load the program initially from the floppy disk. After the program stops loading (and is past checking the copy-protection) you could press the SWAP switch to substitute RAMDrive in place of the floppy device. Any subsequent disk access will be directed to RAMDrive.

Some programs do not access the disk once they have loaded. If a program is copy-protected and has no need of a disk drive after it has been loaded, there is not much chance of using it with RAMDrive. The only method left to try with this type of software is use a memory capture type of cartridge to save the program as an unprotected file. There are also some copy programs (such as Maverick) which will remove protection from programs by using a parameter disk. This is also a good method for making RAMDrive boot-able copies of protected software. Be aware that not all parameters will remove copy protection; many make exact copies with the protection still intact.

Some Specific Applications

While it is impossible to provide specific details on how to use a lot of particular software titles on RAMDrive, there are a couple of alternative operating systems for the C64 and/or C128 which deserve special attention. The majority of applications will require that you experiment with RAMDrive to discover what can and cannot be used.

GEOS

GEOS can be used with RAMDrive, but because GEOS is a very device specific program, special patches are needed to allow GEOS to recognize RAMDrive. These patches have been created by CMD in the form of gateWay, a replacement for the GEOS deskTop. This will allow you to use RAMDrive not only as a normal RAM disk under GEOS, but also allows for the use of some of the standard partition types provided by RAMDrive. Many other benefits are realized by using gateWay, features which go far beyond a minimal level of support for RAMDrive. For more details, see the manual included with gateWay.

CP/M

CP/M may be used to a limited degree with RAMDrive. CP/M contains special drivers for each device it supports. Most of the supported devices are accessed by using burst mode commands. Since RAMDrive is not connected to the serial bus for this type of emulation is not possible; only drivers which have been written to access drives via direct Kernal or DOS calls can be used with RAMDrive. In our testing, only the 1541 driver was found capable of working properly with RAMDrive. When using FORMAT.COM from CP/M, you should format a 1541 partition on RAMDrive using the 128 Single Sided format.

JiffyMON

CMD's JiffyMON V2 machine language monitor program can be used with RAMDrive. Please note that a different SYS command is needed to start JiffyMON when RAMDrive is enabled:

SYS 58564

Section 7
Command Reference

Command Syntax

This section documents many of the RD DOS commands which can be used with on RAMDrive. The syntax for these commands is given in a standard format which should allow you to easily recognize required and optional parameters. Examples are used throughout to assist you in determining proper usage of the commands. If any problem should arise in determining command syntax, be sure to check the following information.

Command String Elements

The command string is made up of a number of elements. In the case of commands sent directly from BASIC, the first part of the command string is usually the command itself. In the case of commands sent via the command channel, the command itself is usually found at the beginning of a string sent to RAMDrive. The elements of the command string as used in this manual are described below:

Literals are characters which must be entered exactly as shown. These will appear as plain text.

User supplied values are those which must be supplied by the user and whose values and type are dependant upon the use of the command. These will appear as *italicized* text.

Optional parameters and **options** are values or literals which need not be included in the command unless the user wishes to specify the option. Often, the optional parameters will be substituted with a certain default when left out of the command string. Optional parameters will appear within [brackets].

Choice parameters allow or require you to choose from more than one parameter to be placed within the command string. Whenever these appear in the syntax of the statement, the choices will be enclosed by {braces} and the individual choices will be separated by a vertical bar character (|). Only one choice parameter may be used in command string.

You may occasionally notice syntax in which one or more elements is followed by three periods (...). This means that the parameter last shown may be repeated. This will normally be discussed in more detail in the text describing use of the command.

Example command string

The following example illustrates the command syntax used throughout this manual:

```
HEADER"partname"[,Iid][,Dn][{ON|,}Udv]
```

In this command string, HEADER is the actual command and is entered literally. The commas (,) and the 'I', 'D', 'ON', and 'U' are literals contained within brackets and are optional. The parameters 'partname', id, n, and dv are user supplied variables. These variables are described in a table following the command syntax. The table for this command string would look like this:

where:	partname	=	the name you wish to appear in the partition header
	id	=	a two character ID for the partition header
	n	=	the partition number you wish to format (0 or 1)
	dv	=	the device number of RAMDrive

Since these parameters are all enclosed in brackets, they are optional. If you decide to use one or more of the optional parameters, notice that they are accompanied by a literal. This literal must be included in the order shown along with the user-supplied variable. Also notice that the 'ON' and a comma are enclosed in braces and are separated by a vertical bar. This means that if you want to supply the unit number, it must be preceded by either the literal 'ON' or by a comma. The full use of this command is shown in the following command string:

```
HEADER"PARTITION 1",IP1,D0,U16
```

Since partition (or drive) zero (0:) is assumed in BASIC 7.0 commands, It would also be possible to shorten this command to:

```
HEADER"PARTITION 1",IP1,U16
```

Paths in Command Strings

Throughout this manual you will see commands which have a [path] shown in the command syntax. Paths are only used when accessing Native Mode partitions, and specify which subdirectory the disk operation is intended for. Here is an example of a command with a path in its syntax:

```
VERIFY"[[n][path]:]filename",dv[,sa]
```

Most of the common commands used on RAMDrive will allow you to include a subdirectory path within the command string, provided that the target file is within a Native Mode partition. This path immediately follows the partition number within the command string and is shown throughout this manual within command syntax descriptions as [*path*]. When including one or more subdirectories within a command string, each subdirectory placed in the command must be bracketed between slash (/) characters, and the final slash must usually be followed by a colon.

For example, if you had a file named COPY located in a subdirectory named UTILITIES in partition number 1, you could load this file with the following command:

```
LOAD"1/UTILITIES/:COPY",16
```

The portion of this command which makes up the path is:

```
/UTILITIES/
```

If you had nested subdirectories, and the file COPY was located in a subdirectory named COPIERS which in turn was located within as subdirectory named UTILITIES in partition 1, you might load this file with:

```
LOAD"1/UTILITIES/COPIERS/:COPY",16
```

The portion of this command which makes up the path is:

```
/UTILITIES/COPIERS/
```

You may be able to shorten this command, depending on which directory and partition you are currently located in. For example, if you are already located within partition 1, and the root directory is your current directory, you could skip the partition number in the command string:

```
LOAD"/UTILITIES/COPIERS/:COPY",16
```

If your current partition is partition 1 and your current directory is UTILITIES, you could simply enter:

```
LOAD"/COPIERS/:COPY",16
```

If your current partition is partition 2 but your current directory in partition 1 is UTILITIES, you could enter:

```
LOAD"1/COPIERS/:COPY",16
```

If your current partition is partition 2 but the current directory in partition 1 is COPIERS, you could enter:

```
LOAD"1:COPY",16
```

There is another syntax which will always allow you to begin your path at the root directory. This is helpful when you are located within a different subdirectory in the same partition. For example, if your current directory is GAMES in partition 1, and you wish to load the COPY program shown in the previous example, you can begin your path with two slashes:

```
LOAD"//UTILITIES/COPIERS/:COPY",16
```

Two slashes placed at the beginning of a subdirectory path indicates that the path must begin at the root directory. If you are in a different partition, or

The semi-colon which appears between the portion of the command in quotes and the character string code is optional. Some commands require that numeric variables or actual numbers be used as command parameters. This is more common with direct access commands. An example is given in the following U1 command:

```
OPEN15,16,15:PRINT#15,"U1";2;0;1;34:CLOSE15
```

The numbers shown following the semi-colons in the command given above could also be expressed as variables. For example:

```
OPEN15,16,15:PRINT#15,"U1";C;D;T;S:CLOSE15
```

If numbers are used in a command, they may be included within the string portion as long as each is separated from the string and the other numbers by a space:

```
OPEN15,16,15:PRINT#15,"U1 2 0 1 34":CLOSE15
```

In this case, the trailing quote is placed after the last number or parameter required by the command. You may optionally place a colon at the end of the command itself, no matter which way the command is used:

```
OPEN15,16,15:PRINT#15,"U1:";2;0;1;34:CLOSE15
OPEN15,16,15:PRINT#15,"U1: 2 0 1 34":CLOSE15
```

Reading Disk Errors

Disk errors are most often detected by reading the command channel. This is done by using either the GET# or INPUT# commands. Using INPUT# is the fastest method of returning error information from RAMDrive. GET# is more commonly used for getting non-error information. Since both the GET# and INPUT# commands must use the BASIC input buffer, they cannot be used in direct mode.

Usually, programs will check for errors immediately after attempting to perform a disk access or disk command. Here is a short program which shows how the error channel is read:

```
10 OPEN15,16,15:INPUT#15,E,E$,T,S:CLOSE15
20 PRINTE,E$,T,S
```

As you can see, four parameters are returned via the command channel when checking for an error. These are, in order: the error number, the error message, the track where the error occurred, and the sector where the error occurred. Many particular errors will not occur at any particular track and sector, in which case the track and sector variables will contain zeroes.

There are occasions where it is more desirable to obtain error data or other information from the command channel one byte at a time. In these instances a program similar to the one that follows could be employed:

```
10 OPEN15,16,15
20 GET#15,E$:PRINTE$;:IFST<>64THEN20
30 CLOSE15
```

If you are using JiffyDOS, it is possible to read the error channel without using a program. This is done by pressing the commercial at (@) key and then <RETURN>.

In the preceding example, we used the status variable to determine when the end of the file was reached. The BASIC status variable ST is quite useful in serial device access, so here is a breakdown of the individual bit values. These definitions apply when the specified bit is set, or equal to one.

BIT	DESCRIPTION OF STATUS	STATUS VALUE
7	DEVICE NOT PRESENT or END OF TAPE	128
6	END OF FILE (EOI)	64
5	CASSETTE CHECKSUM ERROR	32
4	VERIFY ERROR or CASSETTE READ ERROR	16
3	DATA BLOCK TOO LONG (Cassette)	8
2	DATA BLOCK TOO SHORT (Cassette)	4
1	TIME OUT ON LISTENER	2
0	TIME OUT ON TALKER	1

Figure 10-1

The C128 provides another method of checking for disk errors via two reserved BASIC 7.0 variables: DS and DS$. The variable DS returns the error number, while DS$ returns the error message string. These variables can be viewed with a PRINT statement, as shown in the following example:

```
PRINTDS,DS$
```

It is not necessary to print both of these variables at the same time, but the error message string will usually provide help in understanding why the error occurred, and also makes it easier to look up in the error descriptions located in Appendix B of this manual.

Note: DS and DS$ are usually valid only after a BASIC 7.0 disk command.

are not sure which directory is the current directory in the partition you wish to access, it is usually wise to use the double-slash method. Remember to include the partition number if you are in a different partition:

```
LOAD"1//UTILITIES/COPIERS/:COPY",16
```

Subdirectory Paths Using JiffyDOS Commands

The examples given above are shown using the standard Commodore DOS methods of loading files. Since RAMDrive is equipped with JiffyDOS, you may use the JiffyDOS load commands. Here are the above examples converted to their JiffyDOS equivalents:

```
/"1/UTILITIES/:COPY",16
/"1/UTILITIES/COPIERS/:COPY",16
/"/UTILITIES/COPIERS/:COPY",16
/"/COPIERS/:COPY",16
///UTILITIES/COPIERS/:COPY
/1//UTILITIES/COPIERS/:COPY
```

You may also leave out the quotes (") and the comma and device number by using the JiffyDOS <CONTROL><D> function to change the default device. The last two examples illustrate this option and show the only situation under which you will use three slash characters in a row. The first slash character is assumed to be the BASIC LOAD command by JiffyDOS. You may also use any of the other JiffyDOS wedge commands for loading and saving programs in conjunction with subdirectory paths. See your JiffyDOS manual for more information about these commands.

Sending Commands from BASIC

Most of the commands you send to RAMDrive will be from BASIC. This requires very little new knowledge since RAMDrive accepts standard disk drive commands. You should note that in order to maximize your use of RAMDrive, it may be desirable to use BASIC 2.0 or DOS command channel commands when operating RAMDrive on a C128 instead of using BASIC 7.0 commands. This is because BASIC 7.0 places some limitations on the use of partition numbers in its command syntax. Specifically, the drive number parameter in BASIC 7.0 commands may only be represented by a zero (0) or a one (1). Since RAMDrive uses this number as an indication of which partition is to be used for the particular command, and since RAMDrive can have up to 31 partitions, the BASIC 7.0 commands may or may not be able to access the desired partition. Also, because of the way in which BASIC 7.0 sends commands, it is not possible to include subdirectory paths within these commands. If you wish to include subdirectory paths, use the BASIC 2.0 or DOS command channel version of the command instead.

The Command Channel

Many of the commands discussed in this section, require you to send the command via RAMDrive's command channel. Opening a command channel to RAMDrive requires the following BASIC statement:

```
OPENlf,dv,sa
```

where: lf = the logical file number
dv = the device number of RAMDrive
sa = the secondary address

The logical file number can be any number from 1 to 127. Other numbers are legal (128-255) but will cause side effects which are not usually desirable, so it is wise to avoid them.

The device number is the same as the device number currently assigned to RAMDrive. This is set to 16 at the factory, but may be changed to any number from 8 to 29 by using RAM-TOOLS, or set to 8 or 9 by using the SWAP feature.

The secondary address is often referred to as a channel. Secondary addresses 0 through 14 are used to open files, whereas secondary address 15 tells RAMDrive that data sent via this channel should be interpreted as commands and command data. Thus, channel 15 is referred to as the 'command channel'. This example shows how to open the command channel:

```
OPEN15,16,15
```

The command channel may be opened from within a program or in 'direct' mode. Whenever you enter a command to be executed immediately, without preceding it with a line number, it is considered to be entered in BASIC's direct mode. Here is an example of sending a command via the command channel:

```
OPEN15,16,15:PRINT#15,"I":CLOSE15
```

This type of command may also be sent without using the PRINT# command as shown below:

```
OPEN15,16,15,"I":CLOSE15
```

Some commands require that other parameters be sent to RAMDrive in the form of character strings with the CHR$ function. These types of commands can be sent using the PRINT# command. An example of this is the device number change command:

```
OPEN15,16,15:PRINT#15,"U0>";CHR$(10):CLOSE15
```

Partition Numbers in File Names

A partition number may be specified within a filename in place of the drive number. The drive number is the number which can precede the colon (:) in Commodore DOS filenames. Often, this number is not included since it is normally used only with dual drives to determine if the file operation should be directed to drive 0 or drive 1. Whenever this drive number is left out of the command, drive 0 is assumed. These rules also apply to RAMDrive, which contains a single RAM disk with up to 31 partitions. Whenever you wish to perform a command or file operation with the current partition, you may use a 0 or leave out the partition number entirely. However, if your current partition is different than the partition in which you wish to perform the file operation, you must either move to that partition first (with the 'CP' command), or specify the partition number within the filename. The following examples should help to illustrate this:

```
LOAD "1:MCOPY",16
OPEN2,16,2,"3:TESTFILE,S,W"
```

Load commands may be abbreviated if you are using JiffyDOS:

```
/1:MCOPY
```

Partition Numbers in Disk Commands

As described above, partition numbers may be used to replace the drive number in filenames. Partition numbers may also be used in this manner when sending disk commands. In fact, any disk command which allows inclusion of a drive number (with the exception of direct access commands), will also allow you to substitute a partition number in its place. You may use partition numbers when formatting, copying, renaming, scratching, validating, and initializing. This is a large part of what makes RAMDrive so compatible with Commodore DOS.

In the case of direct access commands, the current partition is assigned to the direct access file when when that file is opened. As a result, you must use the 'Change Partition' command to select the desired partition before opening a direct access file. Once the file has been opened, all commands sent to that direct access file will refer to the partition you were in when the file was opened, even if the current partition is changed.

Partition Commands

Many of the commands used on RAMDrive are partition related and, among other things, are used to format, initialize and change partitions.

Creating Partitions

Partitions are created by using the RAM-TOOLS program supplied with RAMDrive. Use of this program is documented in Appendix A.

Creating 1581 Style Sub-partitions

You can create 1581-style partitions within 1581 Emulation Mode partitions on RAMDrive. This type of partition is sometimes referred to as a subdirectory by 1581 users. Because of the way these partitions allocate space, we feel the term subdirectory does not apply. Since RAMDrive is already divided into partitions, and since these 1581 'subdirectories' are nested into 1581 Emulation Mode partitions, we have opted to call them sub-partitions.

Certain limitations apply when creating sub-partitions that are intended to store files. Because of the way physical tracks are handled in Commodore's 1581 DOS, and because sub-partitions must contain header and directory blocks, the minimum size of a sub-partition is 120 blocks. The starting sector must be zero, and the ending sector must be a multiple of 40. Sub-partitions are not allowed to begin on, end on, or contain within themselves track 40. Here is the syntax required to create 1581 style sub-partitions:

```
OPENlf,dv,15:PRINT#lf,"/[n]:partname,"CHR$(st)
CHR$(ss)CHR$(sl)CHR$(sh)",C":CLOSE15
```

where:

lf	=	the logical file number for the command channel
dv	=	the current device number assigned to RAMDrive
n	=	the target 1581 Emulation Mode partition
partname	=	the name of the 1581 sub-partition name to be created
st	=	the starting track of the sub-partition
ss	=	the starting sector of the sub-partition
sl	=	the low byte of the sub-partition size in sectors
sh	=	the high byte of the sub-partition size in sectors

Example:

```
OPEN15,16,15:PRINT#15,"/4:SUB1,"CHR$(1)CHR$(0)
CHR$(160)CHR$(0)",C":CLOSE15
```

The preceding example should be entered as one line. There is no JiffyDOS equivalent for this command since it requires the use of the CHR$ function.

Note: Before you can use a newly-created 1581 sub-partition, you must format it. See 'Formatting 1581 Style Sub-Partitions' for more information.

Deleting Partitions

Partitions are deleted by using the RAM-TOOLS program supplied with RAMDrive. Use of this program is documented in Appendix A.

Deleting 1581 Style Sub-partitions

1581 style sub-partitions are handled quite differently than standard RAMDrive partitions. You can delete a sub-partition by using the DOS or BASIC 7.0 SCRATCH commands. Scratching a 1581 sub-partition is no different than scratching a file, although the consequences may be severe if this is done accidentally. Any files contained within the sub-partition will be lost. See 'Scratching (deleting) Files' for the proper command syntax. Remember to substitute the name of the sub-partition for the filename.

Changing Partitions

You may change from one partition to another by sending the 'CP' (Change Partition) command to RAMDrive via the command channel. The syntax is:

```
OPENlf,dv,15:PRINT#lf,"CPn":CLOSElf
```

where: lf = the logical file number for the command channel
dv = the device number of RAMDrive
n = the partition number you wish to change to (1-31)

Example:

```
OPEN15,16,15:PRINT#15,"CP4":CLOSE15
```

JiffyDOS Example:

```
@CP11
```

The 'C'<SHIFT>'P' command is a variation on the Change Partition command which allows it to be used more easily from within BASIC programs. A shifted 'P' is indicated by the symbol '□'. This command allows you to use a character string to indicate the partition. The syntax is:

```
OPENlf,dv,15:PRINT#lf,"C□"+CHR$(n):CLOSElf
```

where: lf = the logical file number for the command channel
dv = the device number of RAMDrive
n = the partition number you wish to change to (1-31)

Example:

```
OPEN15,16,15:PRINT#15,"C□"CHR$(11):CLOSE15
```


Moving Between 1581 Style Sub-partitions

Since the 1581 Emulation Mode partitions on RAMDrive support 1581 style partitioning (that is to say 'sub-partitioning'), you may use the standard DOS commands to change from one 'sub-partition' to another. The syntax is:

```
OPENlf,dv,15:PRINT#lf,"/[n]:[partname]":CLOSElf
```

where: lf = the logical file number for the command channel
dv = the device number of RAMDrive
n = the partition number of the 1581 Emulation partition
partname = the partition name of the 1581 style 'sub-partition'

If the 1581 Emulation Mode partition is the current partition, the partition number n may be left out. The 'sub-partitions' may be nested within each other, but to access one which is two levels down from the currently selected one, you will have to issue the command twice (once with each of the two sub-partition names). To return to the root (main) directory of the 1581 Emulation Mode partition, issue this command without the sub-partition name. This will also occur if you issue an INITIALIZE command to the 1581 Emulation Mode partition.

Please note that if you exit a 1581 Emulation Mode partition with a 'CP' command and then return to it later with another 'CP' command, you will be placed into whichever sub-partition you were in when you exited.

Example of moving to a different 1581 sub-partition:

```
OPEN15,16,15:PRINT#15,"/4:SUB1":CLOSE15
```

JiffyDOS example:

```
@/4:SUB1
```

Formatting Partitions

The standard Commodore DOS NEW command (not to be confused with the BASIC NEW command) may be used to format partitions on RAMDrive from either BASIC 2.0 or BASIC 7.0. Although RAMDrive will also accept the BASIC 7.0 HEADER command, this command is limited to either the current partition (0) or partition 1.

The DOS NEW commands may be used to delete all files from a partition. It is not necessary to format any partitions on RAMDrive before you begin using them, as they have been pre-formatted at the time of creation. Even when you create new partitions on the system, formatting is performed automatically by RAM-TOOLS. You may wish to format them again, however, to change the header name or disk ID. When using the DOS NEW command, RAMDrive can accept both the long and short versions. The

difference in time is slight, especially in comparison to other disk drives (formatting rarely takes longer than a second or two). The BASIC 2.0 or BASIC 7.0 syntax for the DOS NEW command is given below.

```
OPENlf,dv,15:PRINT#lf,"N[n]:partname[,id]"
:CLOSElf
```

where: lf = the logical file number for the command channel
dv = the current device number of RAMDrive
n = the partition you wish to format (0-31)
partname = the name you wish to appear in the partition header
id = a two character id for the partition header

The following syntax applies to the BASIC 7.0 HEADER command:

```
HEADER"partname"[,Iid][,Dn][{ON|,}Udv]
```

where: partname = the name you wish to appear in the partition header
id = a two character id for the partition header
n = the partition number you wish to format (0 or 1)
dv = the device number of RAMDrive

Examples:

```
OPEN15,16,15:PRINT#15,"N3:PARTITION 3,P3":CLOSE15
HEADER"PARTITION 1",IP1,D1 ON U16
```

JiffyDOS examples for using the DOS NEW command:

```
@N3:PARTITION 3,P3
@"N3:PARTITION 3,P3",16
```

Note: The DOS NEW and BASIC 7.0 HEADER commands will not be accepted if issued from within a Native Mode subdirectory.

Formatting 1581 Style Sub-partitions

The DOS NEW commands are also used to format the 1581 style sub-partitions which may be created within 1581 Emulation Mode partitions. 1581 sub-partitions must be formatted before they can be used. Before attempting to format a sub-partition, you must make sure that the sub-partition is the current sub-partition within the 1581 Emulation Mode partition that contains it. Do this by using the command outlined in 'Moving Between 1581 Style Sub-partitions' elsewhere in this section. To avoid formatting the wrong area on RAMDrive, it is usually wise to make the appropriate 1581 Emulation Mode partition the current partition.

Initializing Partitions

The DOS INITIALIZE command is often used when a different disk is inserted into a floppy disk drive. This ensures that the drive updates its buffer with a copy of the BAM on the new disk. This function is performed automatically by RAMDrive, but the command has been implemented to retain compatibility. Note that initializing a 1581 Emulation Mode partition causes it to return to the root directory (ala the 1581). The syntax is:

```
OPENlf,dv,15:PRINT#lf,"I[n][:]":CLOSElf
```

where: lf = the logical file number for the command channel
dv = the current device number assigned to RAMDrive
n = the partition to be initialized

Example:

```
OPEN15,16,15:PRINT#15,"I3:":CLOSE15
```

JiffyDOS example:

```
@I3:
```

Validating Partitions

The DOS VALIDATE and BASIC 7.0 COLLECT commands check all files in a partition to verify proper allocation of disk space, free any improperly allocated blocks, and delete unclosed (splat '*') files. You should not use these commands on partitions that contain blocks allocated via the BLOCK-ALLOCATE command, or else information may be lost. The VALIDATE command must be used with BASIC 7.0 if you wish to validate a partition other than the current (0) partition or partition 1. The syntax is:

```
OPENlf,dv,15:PRINT#lf,"V[n][:]":CLOSElf
```

where: lf = the logical file number for the command channel
dv = the device number of RAMDrive
n = the partition you wish to validate (0-31)

The syntax for the BASIC 7.0 COLLECT command is:

```
COLLECT[,Dn][{ON|,}Udv]
```

where: n = the partition you wish to validate (0 or 1)
dv = the device number of RAMDrive

Examples:

```
OPEN15,16,15:PRINT#15,"V2:":CLOSE15
COLLECT,D0,U16
```

JiffyDOS example of the DOS VALIDATE command:

```
@V2
```

Partition directory

Having multiple partitions on RAMDrive necessitates having the ability to view a directory of partitions. The partition directory may be viewed while you are working within any partition and relates information concerning the number, name, and type of each partition on RAMDrive. This command also contains options that allow you to specify which partitions will be listed. The syntax for this command is as follows:

```
LOAD"$=P[:*][=tp]",dv
```

where: tp = partition type - N = native
4 = 1541
7 = 1571
8 = 1581
dv = the current device number assigned to RAMDrive

Examples:

```
LOAD"$=P",16
LOAD"$=P:*",16
LOAD"$=P:*=8",16
```

JiffyDOS examples:

```
@$=P
@"$=P:*=N",16
```

Renaming Partitions

If you reformat a partition with the DOS NEW command, you may wish to change its name in the partition directory as well. In order to do this we have added the 'Rename Partition' command. This command is similar to the DOS command which is used to rename files. The syntax is as follows:

```
OPENlf,dv,15:PRINT#lf,"R-P:newname=oldname":CLOSElf
```

where: lf = the logical file number
dv = the current device number of RAMDrive
newname = the name you wish to assign to the partition
oldname = the name of the partition in the partition directory

Example:

```
OPEN15,16,15,"R-P:WORK=NATIVE 1":CLOSE15
```

JiffyDOS Examples:

```
@R-P:WORK=NATIVE 1
@"R-P:WORK=NATIVE 1",16
```


Renaming Directory Headers

After placing a number of files within a given partition or subdirectory, you may wish to change the name that appears in the directory header (displayed at the top of the directory listing for that partition or subdirectory). Although this could be done by using the DOS NEW command, all files within that partition would be lost at the same time. In order to allow you to rename a header without having to lose or copy all of your files, RAMDrive has a 'Rename Header' command. The syntax is as follows:

```
OPENlf,dv,15:PRINT#lf,"R-H[n][path]:newname":CLOSElf
```

where: lf = the logical file number
dv = the current device number of RAMDrive
n = the partition where the header is to be renamed
path = the subdirectory path
newname = the new name for the specified header

Examples:

```
OPEN15,16,15,"R-H:WORK":CLOSE15
OPEN15,16,15,"R-H3:DOWNLOADS":CLOSE15
OPEN15,16,15,"R-H1//ASSEM/:BUDDY64":CLOSE15
```

JiffyDOS Examples:

```
@R-H:WORK
@"R-H//ASSEM/:BUDDY64",16
```

Getting Partition Information

The DOS 'Get Partition Info' command has been created for the purpose of gathering information about the current or some other specific partition. This command will prove valuable to the programmer whose software must react differently to partitions of various types. The partition number for which the information is requested may be placed into a variable and inserted into the command as a character string. The syntax for the G-P command is:

```
OPENlf,dv,15:PRINT#lf,"G-P"[+CHR$(n)]:CLOSElf
```

where: lf = the logical file number for the command channel
dv = the device number of RAMDrive
n = the partition number (0-31)

If the 'G-P' command is sent without the optional character string or a value of 255, the information returned will be for the current partition. A value of 0 requests that the information returned is to be for the system partition. Thirty bytes (0-29) of information concerning the requested partition plus a CHR$(13) are returned over the error channel. The following is a list of this information:

Byte 0	-	Partition type	0	- not created
			1	- Native Mode
			2	- 1541 Emulation Mode
			3	- 1571 Emulation Mode
			4	- 1581 Emulation Mode
			7	- Foreign Mode (Direct Access)
			255	- System
Byte 1	-	CHR$(0) (reserved)		
Byte 2	-	Partition number		
Bytes 3-18	-	Partition name as displayed in the partition directory		
Byte 19	-	Starting system address of partition (high byte)		
Byte 20	-	Starting system address of partition (middle byte)		
Byte 21	-	Starting system address of partition (low byte)		
Bytes 22-26	-	CHR$(0) (reserved)		
Byte 27	-	Size of partition (high byte)		
Byte 28	-	Size of partition (middle byte)		
Byte 29	-	Size of partition (low byte)		
Byte 30	-	CHR$(13)		

Note: The values returned in bytes 19-21 and 27-29 are specified in 256 byte blocks. Also keep in mind that any currently undefined bytes may later be used for specific purposes.

Important: To avoid problems with reading information from Partition 13, the G-P command should always be sent with a trailing carriage return (CHR$(13)). The BASIC PRINT# statement will do this for you automatically as long as you do not follow it with a trailing semicolon (;).

Autobooting

It is possible to autoboot from RAMDrive when it is used with a C128 or 128D in 128 mode. To do so, RAMDrive must either be configured as device number 8, or you must issue the BASIC 7.0 BOOT command. You must also make sure that the current partition has a valid boot sector. The boot sector is located at track 1, sector 0 in all partitions, including Native Mode partitions. An interesting benefit in Native Mode partitions is that the boot sector is always allocated. It is therefore never in danger of being overwritten by files, and cannot be freed by the DOS VALIDATE command. The following syntax applies to the BOOT command:

```
BOOT[[Dn]{ON|,}[Udv]]
```

where: n = the partition where the file is located (0 or 1)
dv = the device number currently in use by RAMDrive

The partition number for this command may only contain a zero or a one. Zero is used to indicate the current partition, while one indicates partition number 1. This command will not accept any other partition numbers, due

to a limitation in the BASIC 7.0 command parsing routines. The structure of the boot sector is the same as found on standard Commodore disk drives.

Examples:

```
BOOT U16
BOOT D0,U16
```

Subdirectory Commands

Three new DOS commands have been added to allow you to create and remove subdirectories, as well as to change the current directory. Both the Create and Change commands use a similar syntax, while the syntax for the Remove command has been limited to avoid problems. These commands, like the subdirectories themselves, are very similar to those found in MS-DOS.

Creating Native Mode Subdirectories

The 'Make Directory' command allows you to create Native Mode subdirectories. This command allows you to use standard path syntax. Using a path allows you to create a subdirectory in any native mode partition no matter what your current partition or current directory may be. Here is that syntax:

```
OPENlf,dv,15:PRINT#lf,"MD[n][path]:name":CLOSElf
```

where: lf = the logical file number for the command channel
dv = the current device number assigned to RAMDrive
n = the Native Mode partition where the subdirectory is to be created
path = the path to the subdirectory in which the new subdirectory will be created
name = the name of the new subdirectory

The above syntax may seem slightly confusing, so here are a few guidelines to help you understand how this syntax works:

1. The name of the subdirectory you wish to create must always be separated from the rest of the command by a colon (:).

2. If you are creating a subdirectory within another subdirectory, you must specify that subdirectory within the path unless it is your current directory.

3. If subdirectories are specified within the path of the command (to the left of the colon), each subdirectory name must fall between slash (/) characters (only 1 slash is needed between subdirectory names).

4. Paths normally start at the current directory. If you want the path to start at the root directory (the main directory in that partition), the path should begin with two slashes.

5. If the subdirectory is to be created in a partition other than the partition in which you are located, place the partition number at the start of the path (in front of any slashes).

The following examples should help clarify these guidelines:

```
OPEN15,16,15:PRINT#15,"MD:TEMP":CLOSE15
OPEN15,16,15:PRINT#15,"MD1:TEMP":CLOSE15
OPEN15,16,15:PRINT#15,"MD1//:TEMP":CLOSE15
OPEN15,16,15:PRINT#15,"MD1//TEMP/:TEMP2":CLOSE15
OPEN15,16,15:PRINT#15,"MD/TEMP/:TEMP2":CLOSE15
```

JiffyDOS Examples:

```
@MD:TEMP
@MD1:TEMP
@MD1//:TEMP
@"MD1//TEMP/:TEMP2"
@"MD/TEMP/:TEMP2",16
```

Moving Between Native Mode Subdirectories

The 'Change Directory' command allows you to move between Native Mode subdirectories. This command employs the same syntax used in the 'Make Directory' command. Using a path allows you to move to a subdirectory anywhere in the currently selected Native Mode partition. This command will also allow you to change the currently selected directory in any other partition, but will not move you into that directory. In order to move to the current directory of a different partition, you must issue a 'Change Partition' command. The 'Change Directory' syntax is:

```
OPENlf,dv,15:PRINT#lf,"CD[n]{[←]|[[path][:]
subname]}":CLOSElf
```

where:
- lf = the logical file number for the command channel
- dv = the current device number assigned to RAMDrive
- n = the Native Mode partition where the subdirectory you wish to make the current directory exists
- path = the subdirectory path leading to the
- subname = the name of the subdirectory

Note that you can include the back arrow immediately after 'CD[n]' to move backwards one directory (to the PARENT). The back arrow cannot be combined with any subdirectory path information. See the examples below.

It is not required that you include the colon before the subdirectory name, as long as the subdirectory name is preceded by a slash.

Here are some examples of the Change Directory command:

```
OPEN15,16,15:PRINT#15,"CD:TEMP":CLOSE15
OPEN15,16,15:PRINT#15,"CD1//:TEMP":CLOSE15
OPEN15,16,15:PRINT#15,"CD1//TEMP/:TEMP2":CLOSE15
OPEN15,16,15:PRINT#15,"CD1←":CLOSE15
OPEN15,16,15:PRINT#15,"CD/TEMP/TEMP2":CLOSE15
```

JiffyDOS Examples:

```
@"CD:TEMP",16
@CD1//TEMP
@CD1//TEMP/TEMP2
@CD1←
@CD/TEMP/TEMP2
```

Deleting Native Mode Subdirectories

The 'Remove Directory' command allows you to delete Native Mode subdirectories. This command does not allow the use of paths in order to avoid problems with removing a subdirectory which is a parent of the directory in which you are located. This command will not allow you to delete a subdirectory which contains any files - you must delete these files first by using the DOS SCRATCH or the BASIC 7.0 equivalent. The following syntax applies to the 'Remove Directory' command:

```
OPENlf,dv,15:PRINT#lf,"RD[n]:subname":CLOSElf
```

where:
- lf = the logical file number for the command channel
- dv = the current device number assigned to RAMDrive
- n = the partition where the subdirectory you wish to remove exists
- subname = the name of the subdirectory you wish to remove

Here are some examples of the Remove Directory command:

```
OPEN15,16,15:PRINT#15,"RD3:TEMP":CLOSE15
OPEN15,16,15:PRINT#15,"RD:TEMP2":CLOSE15
```

JiffyDOS Example:

```
@"RD3:TEMP",16
@RD:TEMP2
```

Viewing Directories

Directories may be viewed by using the following BASIC 2.0 command:

```
LOAD "$",16
```

This command will load the current directory from your RAMDrive (assuming it is set as device number 16). You may then issue the LIST command to view the directory on your screen. You may also use a partition number for selecting the directory to be loaded, as in the following example:

```
LOAD "$2",16
```

Pattern Matching

Selective directories may also be loaded in the standard way, by placing a colon (:) at the end of the partition number or path, and by using a filename or pattern matching characters to determine which files to include in the listing. The equals (=) sign and a filetype designator may also be included after the filename to indicate a particular filetype. The filetype characters are: P for program (PRG), S for sequential (SEQ), U for user (USR), R for relative (REL), and B for subdirectory branch (DIR).

Examples:

```
LOAD "$2:S*=P",16
```

This example will load a directory of all PRG files (=P) starting with an S (S*) from partition number 2. You may also use the asterisk at the beginning of a filename as in the following example:

```
LOAD "$1/UTILS/:*E=P",16
```

This example will load a directory of all PRG files which end with an 'E'. The question mark '?" may also be used to replace an unknown character in the filename. It is also possible to use the asterisk in the middle of a pattern as shown in this example:

```
LOAD "$1/UTILS/:R*E=P",16
```

This pattern will match filenames like RIDE and RUE. Only one asterisk may be used in the pattern. Another matching character is the question mark which will match any character found at that position in the filename.

```
LOAD "$2:B?RE=P",16
```

This example will load a directory of all PRG files which are four characters long, start with 'B' and end with 'RE'. More than one question mark may be used in a pattern. It is also possible to mix question marks and an asterisk together in a pattern.

File Commands

File commands are the most commonly used commands. They include loading and saving files, verifying, renaming, scratching, copying, and locking files and are entered in much same manner as for Commodore or compatible disk drives. Although the BASIC 2.0 versions of these commands are supported in BASIC 7.0, BASIC 7.0 also contains other commands that perform the same functions. Note that the BASIC 2.0 versions of these commands allow more versatility when dealing with partitions.

Loading Files

The following syntax can be used to load programs in BASIC 2.0 and BASIC 7.0:

```
LOAD"[[n][path]:]filename",dv[,sa]
```

where:	n	=	is any legal partition number from 1 to 31
	path	=	the subdirectory path to the file
	filename	=	is any legal filename of up to 16 characters
	dv	=	is the current device number assigned to RAMDrive
	sa	=	is the secondary address if needed

To load a machine language program, a secondary address of 1 must be added to the end of this command, separated from the device number by a comma .

Examples:

```
LOAD "2:BASEBALL",16
LOAD "3/TERMS/:TERMBOOT",16,1
```

JiffyDOS examples:

```
/"2:BASEBALL",16
/2:BASEBALL
%3/TERMS/:TERMBOOT
```

It is generally a good idea to use the BASIC 2.0 syntax if you are specifying partitions since BASIC 7.0 will only allow access to the current (0) partition or partition 1, and will not allow the use of subdirectory paths.

The BASIC 7.0 BLOAD command can be used to load machine language or data files into memory. The DLOAD command is used primarily to load BASIC programs. The syntax for these commands is shown below.

```
BLOAD"filename"[,Dn][{ON|,}Udv][,Bb][,Pa]

DLOAD"filename"[,Dn][{ON|,}Udv]
```

where: filename = the name of the machine language program to be loaded
n = the partition number where the file is located (0 or 1)
dv = the device number currently in use by RAMDrive
b = the memory bank where the file is to be loaded
a = the starting address of the file to be loaded

Examples:

```
BLOAD"SPRITE",D0,U9,B0,P3584
DLOAD"TEST",D0 ON U9
DLOAD"TEST2"
```

Saving Files

The following syntax can be used to save programs in BASIC 2.0 and BASIC 7.0:

```
SAVE "[[@][n][path]:]filename",dv
```

where: n = is any legal partition number from 1 to 31
path = the subdirectory path where the file is to be saved
filename = is any legal filename of up to 16 characters
dv = is the current device number assigned to RAMDrive

The '@' symbol shown in the command syntax may be used to indicate that a file with the same name which already exists should be replaced with the new file. This is called the 'Save with Replace' option and if it is used, it must be followed by a partition number and a colon (:). To save a machine language program from a C64 or C128 in 64 mode, you must use a machine language monitor or change some of the BASIC pointers.

Examples:

```
SAVE"2:BASEBALL",16
SAVE"/TERMS/:TERMBOOT",16
```

JiffyDOS examples:

```
←"2:BASEBALL",16
←/TERMS/:TERMBOOT
```


You may use the BASIC 7.0 BSAVE and DSAVE commands when it is your intention to work with your current partition (0) and directory, or in the current directory of partition 1. BSAVE is intended for files other than BASIC programs, while DSAVE is intended for BASIC programs.

```
BSAVE"[@]filename"[,Dn][{ON|,}Udv][,Bb],Pa TO Pe

DSAVE"[@]filename"[,Dn][{ON|,}Udv]
```

where: filename = the name of the file to be saved
n = the partition number where the file is to be saved
dv = the device number currently in use by RAMDrive
b = the memory bank where the file is to be saved
a = the starting address of the file to be saved
e = the ending address of the file to be saved

The '@' symbol may be included to indicate that the file being saved is to replace an existing file with the same. This is called 'Save with Replace'.

Examples:

```
BSAVE"SPRITE",D0,U9,B0,P3584
DSAVE"TEST",D1 ON U9
DSAVE"TEST2"
```

Verifying Files

BASIC 2.0 and 7.0 contain commands which allow you to verify if a program has been saved properly. These commands compare the saved program with the contents of memory. Keep in mind that any change in the contents of memory may cause a verify operation to fail. It is best to verify a file immediately after saving it for this reason. Both versions of BASIC support specifying a partition within the filename portion of this command (as described earlier). The following syntax applies to these commands:

```
VERIFY"[[n][path]:]filename",dv[,sa]
```

where: n = the partition where the file is located
path = the subdirectory path to the file you wish to verify
filename = the name of the file to be verified
dv = the current device number assigned to RAMDrive
sa = secondary address of 1 to verify a non-BASIC file

Examples:

```
VERIFY"NEWSTATS",16
VERIFY"1/UTILS/TERMS/:XLATOR",16
```

JiffyDOS example:

```
'"NEWSTATS",16
```

In BASIC 7.0, the standard VERIFY command is accepted, but you may also use the DVERIFY command. This command is limited to use with the current partition (0) or partition 1, and the current subdirectory .

```
DVERIFY"filename"[,Dn][{ON|,}Udv]
```

where: filename = the name of the file to be verified
n = the partition where the file resides (0 or 1)
dv = the device number of RAMDrive

Example:

```
DVERIFY"NEWSTATS",D1,U16
```

Renaming Files and Subdirectories

Filenames and Native Mode subdirectory names may be changed by using either the DOS RENAME or the BASIC 7.0 RENAME command. The BASIC 7.0 version only supports the current directory within the current partition (0) or partition 1. When using either version, the partitions specified for the two file names must either be the same, or must indicate the same partition. The syntax for the DOS RENAME command is:

```
OPENlf,dv,15:PRINT#lf,"R[n][path]:newname=[[n]
[path]:]filename":CLOSElf
```

where: lf = the logical file number for the command channel
dv = the current device number of RAMDrive
n = the partition where the file to be renamed is located
path = the subdirectory path to the file you want to rename
newname = the new name to be assigned to the file
filename = the name of the file which is being renamed

Examples:

```
OPEN15,16,15:PRINT#15,"R1:BOOT1=BOOT":CLOSE15
OPEN15,16,15,"R1/UTILS/:NEWT=1/UTILS/:WW":CLOSE15
```

JiffyDOS Example:

```
@"R1:BOOT1=BOOT",16
```

The BASIC 7.0 RENAME command syntax is:

```
RENAME[Dn,]"filename"TO[Dn]"newfile"[,Udv]
```

where: n = the partition where *filename* is located
filename = the name of the file which is being renamed
newfile = the new name to be assigned to the file
dv = the current device number of RAMDrive

Scratching (deleting) Files

The standard DOS and BASIC 7.0 SCRATCH commands may be used to delete files from a partition. As with many of the other BASIC 7.0 commands, SCRATCH is only effective with partition numbers 0 (current) and 1. The standard Commodore DOS scratch may be used in place of the BASIC 7.0 version when necessary and with BASIC 2.0. The following command syntax covers the DOS version of this command:

```
OPENlf,dv,15:PRINT#lf,"S[n][path]:filename[,[n]
[path]:]filename...]":CLOSElf
```

where: lf = the logical file number for the command channel
dv = the current device number of RAMDrive
n = the partition(s) which hold the file(s) to be scratched
path = the subdirectory path(s) to the file(s)
filename = the name of the file(s)

Multiple files may be scratched with this command which will accept up to five separate filename parameters. Different partitions can be specified with the separate filenames. The filename parameters may also contain wildcards to allow scratching of multiple files within a single partition.

Examples:

```
OPEN15,16,15:PRINT#15,"S1:JUNK,3:C?*.BAS":CLOSE15
OPEN15,16,15,"S1/UTILS/:CO*":CLOSE15
```

JiffyDOS Examples:

```
@"S1:JUNK,3:C?*.BAS",16
@S1/UTILS/:CO*
```

The BASIC 7.0 SCRATCH command syntax is:

```
SCRATCH"filename"[,Dn][{ON|,}Udv]
```

where: filename = the name of the file to be scratched
n = the partition where the file to be scratched resides
dv = the device number of RAMDrive

Multiple files may also be scratched with this command by using pattern matching, although it does not allow you to specify multiple file names as does the DOS version. Remember, this command is only valid for use with the current directory in partition numbers 0 (current) or 1.

Copying Files

Copying is an important consideration with any storage device. For this reason RAMDrive comes supplied with DOS commands which allows you to copy files between partitions. This function may also be accomplished by using one of the copy programs supplied with RAMDrive. Another copy program supplied with RAMDrive allows you to copy an entire disk to a similar partition on RAMDrive and vice versa. For more information on the copy programs included with RAMDrive, see Appendix A.

Copying files between RAMDrive and other drives

Files may be copied between RAMDrive and other disk drives using a standard file copier. Only generic copiers that do not try to discover the drive type by checking ROM locations will work with RAMDrive.

We have included FCOPY with RAMDrive to assist in file copying. FCOPY will work with all file types and all drive types including an REU running under RAMDOS.

Since RAMDrive has JiffyDOS included, you may use the built-in JiffyDOS file-copier with RAMDrive as well. Many of the commercial copy programs will not work with RAMDrive because they look at specific memory locations to try to identify the drive type being used, or attempt to write drive specific code into the disk drive to speed up the copy process.

Copying and Combining files between partitions

You may copy files from one partition to another on RAMDrive by using the standard Commodore DOS COPY command. This command allows you to place a partition number in front of each of the filenames specified in the command. The syntax for this command is as follows:

```
OPENlf,dv,15:PRINT#lf,"C[n][path]:newfile=[[n]
[path]:]filename[,[n:]filename...]":CLOSElf
```

where:
- lf = the logical file number for the command channel
- dv = the current device number of RAMDrive
- n = the partition which holds or is to receive the file
- path = the subdirectory path(s) where the file(s) to be copied or created is (are) located
- newfile = the name of the new file being created
- filename = the name of the file(s) which is (are) being copied

Up to five files may be combined into a single file by using this command, though it is important to note that copying a number of files into a single file is only effective with text files. If you use this command for copying a single file from one file to another file in a different partition, you may use the same filename for both files.

Examples:

```
OPEN15,16,15:PRINT#15,"C1:FCOPY=3:FCOPY":CLOSE15
OPEN15,16,15,"C:FULLSTATS=STAT1,3:STAT3":CLOSE15
OPEN15,16,15,"C2:MCOPY=1/COPIERS/:MCOPY":CLOSE15
```

JiffyDOS Examples:

```
@"C1:FCOPY=3:FCOPY",16
@"C2:MCOPY=1/COPIERS/:MCOPY"
@C:FULLSTATS=STATS1,3:STATS3
```

You may also copy files from one partition to another on RAMDrive by using the standard BASIC 7.0 COPY command. This command is limited to copying files in the current directory of the current partition (0) or partition 1. Use the DOS COPY command mentioned earlier if you want to copy files between other partitions. The syntax for this command is as follows:

```
COPY[Dn,]"filename"TO[Dn]"newfile"[,Udv]
```

where:
- n = the partition which holds or is to receive the file
- filename = the name of the file(s) which is being copied
- newfile = the name of the file being created
- dv = the current device number of RAMDrive

If you use this command to copy a file to a different partition, you may use the same filename for both files. If they are to reside in the same partition, you must use different filenames or an error will result.

Two files may be combined into a single file by using the BASIC 7.0 CONCAT command, although it is important to note that adding files together in this manner is only effective with text files. The BASIC 7.0 limitation of using only the current directory of the current partition (0:) or partition 1 applies to this command. The syntax is:

```
CONCAT[Dn,]"filename"TO[Dn]"newfile"[,Udv]
```

where:
- n = the partition where the file exists
- filename = the name of the file which is being added
- newfile = the name of the file being added to
- dv = the current device number of RAMDrive

The file previously named *newfile* will be replaced by the newly created combined file. The only way these files may have the same name is if they exist in different partitions.

Example:

```
CONCAT "NEWNUMBERS" TO "ALLNUMBERS"
```

Locking and Unlocking Files

Files located on RAMDrive may be locked to avoid scratching them by accident. Before you can scratch a locked file, it must first be unlocked. Locking a file sets one of the bits in the filetype byte for that file (located in the directory entry of that file). Files which have been locked will appear in the directory with a 'less-than' symbol to the right of the filetype. For example, a file named JIFFYMON which has been locked will appear in the directory listing as:

```
33    "JIFFYMON"          PRG<
```

It is also possible to lock Native Mode subdirectories and 1581 sub-partitions. If a subdirectory has been locked, it is not possible to delete it with the 'Remove Directory' command until it has been unlocked.

The Lock command is a 'toggle' function. Using it on an unlocked file will cause the file to become locked. Using it on a file which has already been locked will unlock that file. The syntax for locking and unlocking files is:

```
OPENlf,dv,15:PRINT#lf,"L[n][path]:name":CLOSElf
```

where:
- lf = the logical file number
- dv = the current device number of RAMDrive
- n = the partition number in which the file exists
- path = the Native Mode subdirectory path in which the file exists
- name = the name of the file or subdirectory you wish to lock or unlock

The following examples illustrate the use of this command:

```
OPEN15,16,15:PRINT#15,"L:TEST":CLOSE15
OPEN15,16,15:PRINT#15,"L1//:TEST":CLOSE15
OPEN15,16,15:PRINT#15,"L/UTILS/:TEST":CLOSE15
```

JiffyDOS examples:

```
@L:TEST
@"L1//:TEST"
@"L/UTILS/:TEST",16
```

Note: JiffyDOS contains its own version of the LOCK command. This may also be used with RAMDrive. See your JiffyDOS manual for details on using this version.

Relative File Commands

Relative files are files which contain an index table to allow quicker access to a particular portion of the file called a record. Records are kept track of by a special section of the relative file called a side sector. Two different types of side sectors exist in RAMDrive: Regular side sectors (the default type used in 1541 and 1571 Emulation Mode partitions), and super side sectors (the default in 1581 Emulation Mode and Native Mode partitions).

Relative files may be up to 720 blocks long when regular side sectors are used. Relative files which use a super side sector may be over 60,000 blocks long. Up to 65,535 records are allowed and each record may be from 2 to 254 characters in length. All records in the same file are the same length, although it is not necessary to use all of the characters in each record.

The main advantage of using a relative file is speed. Normally, you must read from the beginning of a file until the information you wish to find is reached. With relative files, as long as you keep track of which record your information is stored in, you may go directly to that area of the file and read or write the required information right away. A separate index is usually kept in another file type to keep track of the particular information that each record contains.

Opening or Creating a Relative File

The same syntax can be used to create new or open existing relative files by using the BASIC 2.0 OPEN command or the BASIC 7.0 DOPEN command. When creating a new relative file, print a new record to the file after opening it, using the PRINT# command. When you are done accessing the relative file, close it with the CLOSE command, or the DCLOSE command if you used the BASIC 7.0 syntax to open it. Here is the BASIC 2.0 syntax for opening or creating a new relative file:

```
OPENlf,dv,sa,"[[n][path]:]filename[{"|,L,"
+CHR$(rl)}]
```

where:
- lf = the logical file number
- dv = the current device number of RAMDrive
- sa = the secondary address (2 through 14)
- n = the partition in which the file exists or is to be created
- path = the Native Mode subdirectory path which leads to the file
- filename = the name of the relative file
- rl = the record length (only needed when creating a new relative file)

Examples:

```
OPEN2,16,2,"1/DATA/:CUSTOMERS,L,"+CHR$(127)
OPEN2,16,2,"ADDRESS"
```

This syntax may also be used in BASIC 7.0, but an alternate syntax was provided for this newer version of BASIC. Again, as with all BASIC 7.0 specific commands, you will be limited to using the current partition (0) or partition 1, and you will not be able to specify subdirectories. Here is that alternate syntax:

```
DOPEN#lf,"filename"[,Lrl][,Dn][,Udv]
```

where:
- lf = the logical file number
- filename = the name of the relative file
- rl = the record length (only needed when creating a new relative file)
- n = the partition in which the file exists or is to be created (only 0 or 1 is accepted)
- dv = the current device number of RAMDrive

Examples:

```
DOPEN#2,"CUSTOMERS",L127,D1,U16
DOPEN#2,"ADDRESS"
```

Positioning to a Specific Record

When you are ready to read or write a specific record, it is necessary to use either the DOS POSITION command, or the BASIC 7.0 RECORD command. The relative file must already be opened in order to use these commands, and in the case of the DOS POSITION command, you must also have the command channel open to RAMDrive. These commands may also be used to create a number of blank records when first creating a relative file. This makes writing the actual data much faster. Here is the syntax for the DOS POSITION command:

```
PRINT#lf,"P"+CHR$(ch)+CHR$(lr)+CHR$(lh)[+CHR$(of)]
```

where:
- lf = the logical file number for the command channel
- ch = the secondary address used when opening the relative file plus a value of 96
- lr = the low byte of the record number you wish to access or create
- lh = the high byte of the record number you wish to access or create
- of = the byte number in the record which you wish to start reading from or writing to (first byte if left out)

Examples:

```
PRINT#15,"P"+CHR$(98)+CHR$(30)+CHR$(0)+CHR$(10)
PRINT#15,"P"+CHR$(98)+CHR$(30)+CHR$(0)
```

Here is the syntax for the BASIC 7.0 RECORD command:

```
RECORD#lf,rn[,of]
```

where:
- lf = the logical file number for the relative file
- rn = the record number you wish to access or create
- of = the byte number in the record which you wish to start reading from or writing to (first byte if left out)

Examples:

```
RECORD#2,30,10
RECORD#2,30
```

If you are creating a relative file ahead of time, use the PRINT# command to print a CHR$(255) to the last record. This character has a special meaning in relative files - it is used as the first character for empty records. You may expand the size of the relative file later if you run out of records simply by writing a record with a higher number than currently exists in the file. It just takes a little longer to write a record which has not been previously created.

Whenever a new record is created, an error will occur in RAMDrive. This is error number 50 which means "RECORD NOT PRESENT". Obviously, if it was your intent to create this record, this error can be ignored.

Note: Although it is not necessary to send the RECORD or POSITION commands twice on RAMDrive to avoid data corruption, this practice should be followed anyway to avoid problems when using other drives with your program. To do this, send the RECORD or POSITION command once before writing a record as you normally would, and once again afterward. This will help to ensure that your data will not be corrupted due to the flaw which exists in other disk drives.

Special RAMDrive Commands

Some additional commands have been provided in RD DOS to allow you to take advantage of special features built into this unique system.

Software SWAP Commands

RAMDrive allows you to perform the SWAP 8 and SWAP 9 functions from within software in addition to using the front panel switches. You may also undo any currently active SWAP condition with a SWAP TO DEFAULT command. These commands have been included to allow loaders or boot programs to easily swap RAMDrive's device number.

Whenever one of the SWAP commands is used within a program, it is very important to include a delay loop after issuing the command, even before you attempt to close the command channel. Since these commands instruct RAMDrive to send commands to another drive on the system via the serial bus, any attempt by the computer to use the serial bus while this is taking place could cause a bus collision. Although we have found that delay values of 60 or higher within a FOR/NEXT loop will work with most systems, it is recommended that you use a larger value (500 would be good) to leave ample time for these commands to complete their tasks.

Note that the SWAP commands will not operate under certain conditions, usually when there is a file open to a serial bus device In this case, it would be unwise to perform a SWAP anyway. To avoid these kinds of problems, make sure that you send the SWAP command only when no other files (with the exception of the command channel) are open.

Sending a SWAP TO DEFAULT command is not required when you wish to change from a SWAP 8 condition to a SWAP 9 condition, or vice-versa (RAMDrive will swap directly from 8 to 9 or from 9 to 8). SWAP TO DEFAULT is only needed when you wish to return RAMDrive to it's default device number. It is also possible to send the swap commands from direct mode. You may also abbreviate the command if you are using JiffyDOS. The following syntax applies to the SWAP commands:

```
OPENlf,dv,15:PRINT#lf,"S-x":
FORt=1TO500:NEXT:CLOSElf
```

where:
- lf = the logical file number
- dv = the current device number of RAMDrive
- x = 8 to swap with device number 8
9 to swap with device number 9
D to return RAMDrive to its original device number
- t = a variable to be used for the timing loop

Here is a sample program using all of the SWAP commands. This program assumes that RAMDrive is device number 16 at the time the program begins.

```
100 OPEN15,16,15
110 PRINT#15,"S-8":FORI=1TO500:NEXT:CLOSE15
120 OPEN15,8,15
130 PRINT#15,"S-9":FORI=1TO500:NEXT:CLOSE15
140 OPEN15,9,15
150 PRINT#15,"S-D":FORI=1TO500:NEXT:CLOSE15
```

Direct mode example:

```
OPEN15,16,15,"S-8"
CLOSE15
```

Note: Using these commands in direct mode will not require the timing loop as long as you send the SWAP command first, and then close the channel on a second line. By the time you finish typing the CLOSE command, the SWAP function will be done using the serial bus.

JiffyDOS examples:

```
@S-8
@"S-9",8
```

Direct Access Commands

Most direct access commands require that files be opened to both the command channel and to a direct access file. Before using the commands described in this section, you should be familiar with the methods required to open and access the command channel and a file data channel.

The Direct Access Channel

Opening a direct access channel requires the following BASIC statement:

```
OPENlf,dv,sa,"#[bu]"
```

where:
- lf = the logical file number
- dv = the device number of RAMDrive
- sa = the secondary address
- bu = the buffer number to be used

The logical file number can be any number from 1 to 127, but cannot be the same any logical file currently in use.

The device number is the one currently assigned to RAMDrive (8-29).

The secondary address (also referred to as the channel number) may be any number from 2 to 14. The channel number should be different than any other currently active channel number on RAMDrive. It is usually a good practice to use the same number for the logical file number and the channel number. This usually makes keeping track of files easier for the programmer.

The buffer number may currently be any number from 0 through 7. This is an optional parameter, and if left out, RAMDrive will automatically assign the next available buffer. It is usually best to leave the buffer number out unless you need to use a specific one. If you select the buffer number yourself, be sure to check for an error, since selecting a buffer which is already in use will produce an error condition. This example show how to open a direct access file:

```
OPEN2,16,2,"#"
```

Note: Direct access files are always opened to the current partition number on RAMDrive. If you wish to open a direct access file to a partition other than the one you are currently in, you must change partitions with the 'CP' command first. All further access to this file will occur in that partition number, even if you change partitions after opening the file.

Reading and Writing Data with Direct Access

When BLOCK-READ or BLOCK-WRITE commands are being used, information is normally read or written using the BASIC commands GET#, INPUT#, and PRINT#. The data being accessed with these commands is not being written to or read from the disk directly, but is actually stored in a buffer which temporarily holds the data for a particular block on disk. This may seem a little confusing to those who have never used these commands before, so here is a simpler way of looking at the process.

Reading Data from RAMDrive

If you wish to read data directly from the RAM disk, a BLOCK-READ or U1 command is sent to RAMDrive to tell it to read a particular block from the RAM disk. This block is placed into a 256-byte RAM buffer in RAMDrive. To transfer this data from RAMDrive to the computer, the BASIC commands GET# or INPUT# are used. If you only need a portion of the data in the block, you can use the BUFFER-POINTER command to tell RAMDrive which byte will be the first to be passed to the computer with GET# or INPUT#.

Here is a quick example of reading the seventh byte from track 1 sector 0 of device number 16. The buffer points to byte 6 because the byte numbers start at 0, so byte 6 is actually the seventh byte.


```
10 OPEN15,16,15:REM OPEN COMMAND CHANNEL
20 OPEN2,16,2,"#":REM DIRECT ACCESS CHANNEL
30 PRINT#15,"U1";2;0;1;0:REM TRACK 1 SECTOR 0
40 PRINT#15,"B-P";2;6:REM SEVENTH BYTE (6)
50 GET#2,A$:REM READ THE BYTE INTO A$
60 CLOSE2:REM CLOSE DIRECT ACCESS CHANNEL
70 CLOSE15:REM CLOSE COMMAND CHANNEL
```

If you know which buffer is being used, a MEMORY-READ command could also be used for this purpose. Here is an example of this method:

```
10 OPEN15,16,15:REM OPEN COMMAND CHANNEL
20 OPEN2,16,2,"#2":REM USE BUFFER #2
30 PRINT#15,"U1";2;0;1;0:REM TRACK 1 SECTOR 0
40 PRINT#15,"M-R"CHR$(6)CHR$(5):REM LOC $0506
50 GET#15,A$:REM READ THE BYTE INTO A$
60 CLOSE2:REM CLOSE DIRECT ACCESS CHANNEL
70 CLOSE15:REM CLOSE COMMAND CHANNEL
```

Writing Data to RAMDrive

If you wish to write data directly to the disk, the PRINT# command is first used to write this data to a 256-byte buffer in RAMDrive. After the data has been written to this buffer, the buffer itself is then transferred to the desired track and sector location on the RAM disk by using the BLOCK-WRITE or U2 command.

Here is a quick example of writing to the seventh byte on track 1 sector 0 of device number 16. As in the previous example, the buffer points to byte 6 because the byte numbers start at 0, so byte 6 is actually the seventh byte.

```
10 OPEN15,16,15:REM OPEN COMMAND CHANNEL
20 OPEN2,16,2,"#":REM DIRECT ACCESS CHANNEL
30 PRINT#15,"B-P";2;6:REM SEVENTH BYTE (6)
40 PRINT#2,CHR$(65):REM WRITE BYTE TO BUFFER
50 PRINT#15,"U2";2;0;1;0:REM TRACK 1 SECTOR 0
60 CLOSE2:REM CLOSE DIRECT ACCESS CHANNEL
70 CLOSE15:REM CLOSE COMMAND CHANNEL
```

The above example writes a sector to the disk with no regard for what existed in that sector previously. If you want to change a byte in the sector without changing the rest of the contents of that sector, read it into the buffer first, change the desired byte to the new value, and then write the sector back to disk.

If you know which buffer is being used to write the data, a MEMORY-WRITE command might also be used for this purpose. Here is an example of this method:

```
10 OPEN15,16,15:REM OPEN COMMAND CHANNEL
20 OPEN2,16,2,"#2":REM USE BUFFER #2
30 PRINT#15,"M-W"CHR$(6)CHR$(5)CHR$(1)CHR$(65)
:REM LOC $0506, ONE BYTE, VALUE 65
40 PRINT#15,"U2";2;0;1;0:REM TRACK 1 SECTOR 0
50 CLOSE2:REM CLOSE DIRECT ACCESS CHANNEL
60 CLOSE15:REM CLOSE COMMAND CHANNEL
```

Again, this example writes the new sector to the disk without regard to what existed in the sector previously. If you wish to change a byte without changing the rest of the contents in a sector, read the sector into the buffer first, change the byte, and then re-write it to the disk.

Block Commands

The BLOCK commands allow you to read, write, allocate, and de-allocate the sectors on the disk. When using these commands, you should keep in mind which type of partition you are working with, since each partition has different legal track and sector values, with the directory, BAM, and header information stored in different areas. Many of the functions performed with the BLOCK commands may also be performed with the USER commands, and in fact, the USER commands are often preferred since they do not exhibit some of the side effects of the BLOCK commands.

Allocating Blocks

The BLOCK-ALLOCATE command is used to allocate a block directly. When saving or writing files, this function is handled automatically by the DOS. BLOCK-ALLOCATE is normally used after information has been written to disk by a BLOCK-WRITE command. It can also be used to determine the next higher unallocated block available, if you try to allocate a block which has already been allocated. Upon reading the error channel, you will find that the track and sector variables of the error message will contain the track and sector of the next available block. The syntax for the BLOCK-ALLOCATE command is:

```
PRINT#lf,"B-A:";n;t;s
```

where:
- lf = the logical file number used for the command channel
- n = the partition (0) in which the block to be allocated exists
- t = the track on which the block to be allocated exists
- s = the sector of the block to be allocated

Important Note: The bugs that existed with some Commodore drives with the BLOCK-ALLOCATE command do not exist with the RAMDrive version of this command. There were two different bugs associated with this command. One of these caused the entire track (instead of just one sector) to be allocated on 1541 and 1571 disk drive units. The other bug caused the

BAM on 1581 disk drives to be improperly allocated if another block command (such as B-W) was issued after a block allocate. Closing the file or issuing another type of disk command usually got around this problem, but these steps are unnecessary when using RAMDrive.

Freeing Blocks

The BLOCK-FREE command allows you to free a block which is currently allocated. When scratching files, this function is handled automatically by the DOS. BLOCK-FREE is normally used to free a block in which information had been previously placed by a BLOCK-WRITE command, but which is no longer needed. The syntax for the BLOCK-FREE command is:

```
PRINT#lf,"B-F:";n;t;s
```

where:
- lf = the logical file number used for the command channel
- n = the partition (0) in which the block to be allocated exists
- t = the track on which the block to be allocated exists
- s = the sector of the block to be allocated

The Buffer Pointer

The BUFFER-POINTER command allows you to point to a specific byte within a sector which is to be read from or written to. This action occurs within the disk buffer which holds the data for the particular. The syntax for this command is:

```
PRINT#lf,"B-P";ch;pt
```

where:
- lf = the logical file number used for the command channel
- ch = the channel number used for the direct access file
- pt = the byte number within the block you wish to be used as the first byte of the next GET#, INPUT#, or PRINT# statement

The syntax given above assumes that the command and direct access channels have already been opened. The channel number is the same as the secondary address you used when opening the direct access channel. When using the BUFFER-POINTER command, remember that bytes are numbered from 0 to 255. If you need to access the first byte, it is byte 0.

Reading Blocks

The BLOCK-READ command, due to a quirk in the way it operates, is rarely used. Instead, most applications use the 'U1' command, which is very similar to the BLOCK-READ command. The syntax is:

```
PRINT#lf,"B-R";ch;n;t;s
```

where:
- lf = the logical file number used for the command channel
- ch = the channel number used for the direct access file
- n = partition number (always 0)
- t = the track from which you wish to read a block
- s = the sector which you wish to read from

The channel number specified above is the same as the secondary address used to open the direct access file.

The partition number should always be 0 (zero). This is because direct access commands will always access the partition that was the current partition at the time the direct access channel was opened.

The problem with the BLOCK-READ command is that it will not read an entire block as data. Instead, it uses the first byte in the block as a counter for how many bytes are to be read. This allows you to read the full block only if the first byte of the block contains a value of 255. Since most blocks contain a link to the next sector in the first byte, any blocks intended to be read in this fashion must be written with this same consideration in mind.

Writing Blocks

The BLOCK-WRITE command, due to a quirk in the way it operates, is rarely used. Instead, most applications use the 'U2' command, which is nearly identical to the BLOCK-WRITE command. The syntax for the BLOCK-WRITE command is:

```
PRINT#lf,"B-W";ch;n;t;s
```

where:
- lf = the logical file number used for the command channel
- ch = the channel number used for the direct access file
- n = partition number (always 0)
- t = the track to which you wish to write a block
- s = the sector which you wish to write to

The channel number specified above is the same as the secondary address used to open the direct access file.

The partition number should always be 0 (zero). This is because direct access commands will always access the partition that was the current partition at the time the direct access channel was opened.

The problem with the BLOCK-WRITE command is that it will not write an entire block as data. Instead, it uses the first byte in the block as a counter for how many bytes are to be written. This will allow you to write (and later read) the full block only if the first byte of the block contains a value of 255.

Block Execute

The BLOCK-EXECUTE command has not been fully implemented in RAMDrive. If RAMDrive receives a BLOCK-EXECUTE command, it will not create and error, but will simply return without executing the command.

Memory Commands

The memory commands allow you to read from or write to RAMDrive system memory. The memory commands are useful for many applications, but should be handled carefully to avoid corrupting data or important locations used by the DOS.

Reading from RAMDrive System Memory

The MEMORY-READ command allows you to read directly from system memory locations within RAMDrive. This command is similar to the BASIC PEEK command, with the difference being that MEMORY-READ returns the contents of a serial device's memory instead of computer memory. Reading memory is useful in many applications such as determining the type of device being used as a particular device number. The syntax for MEMORY-READ is:

```
PRINT#lf,"M-R"CHR$(ml)CHR$(mh)CHR$(nb)
```

where:
- lf = the logical file number used for the command channel
- ml = the low byte of the starting memory address
- mh = the high byte of the starting memory address
- nb = the number of bytes to be read

The value for the number of bytes to be read can be from 0 to 255. Since it is improbable that you would want to read zero bytes from RAMDrive's system memory, and it is also likely that you may wish to read 256 bytes on many occasions, the value of zero has been altered to mean 256.

After a MEMORY-READ command is sent, the specified bytes are returned over the error channel. Thus, you may use the GET# command to read these bytes one at a time.

To determine the low and high bytes of a decimal address, you can use the following formula:

```
HB=INT(AD/256):LB=AD-HB*256
```

where:
- AD = the decimal address you wish to begin reading from
- HB = the high byte of the starting memory address
- LB = the low byte of the starting memory address

Here is an example program which determines if the device being accessed is a RAMDrive unit:

```
10 OPEN15,16,15
20 PRINT#15,"M-R"CHR$(160)CHR$(254)CHR$(6)
30 FORI=1TO6:GET#15,B$:A$=A$+B$:NEXT
70 CLOSE15
80 IFA$="CMD RD"THENPRINT"RAMDRIVE PRESENT":END
90 PRINT"NOT A RAMDRIVE UNIT":END
```

Note: The MEMORY-READ command will not read past a page boundary.

Writing to RAMDrive System Memory

The MEMORY-WRITE command allows you to write to memory locations within RAMDrive. Using this command is similar to using the POKE command in BASIC, the difference being that MEMORY-WRITE writes to RAMDrive system memory instead of computer memory. MEMORY-WRITE is useful for placing data directly into one of the buffers, the job queue, or other significant memory locations within RAMDrive. The syntax for MEMORY-WRITE is:

```
PRINT#lf,"M-W"CHR$(ml)CHR$(mh)CHR$(nb)CHR$(d)...
```

where:
- lf = the logical file number used for the command channel
- ml = the low byte of the starting memory address
- mh = the high byte of the starting memory address
- nb = the number of bytes to be written
- d = value of the data byte to be written (if more than one byte is to be written use additional CHR$ statements or a string variable)

The value for the number of bytes to be written can range from 1 to 127. When sending a MEMORY-WRITE command, the bytes to be written are placed at the end of the command. This may be done in the form of CHR$ statements, or if more room is needed, as a string variable.

To determine the low and high bytes of a decimal address, you can use the formula given in the MEMORY-READ command above.

Here is an example program which illustrates the use of the MEMORY-WRITE command:

```
10 FORI=1TO127
20 A$=A$+CHR$(I)
30 NEXT
40 OPEN15,16,15
50 PRINT#15,"M-W"CHR$(0)CHR$(8)CHR$(127)A$
60 CLOSE15
70 END
```


Memory Execute

The MEMORY-EXECUTE command has not been fully implemented in RAMDrive. Sending this command to RAMDrive will not cause an error to occur, RAMDrive will simply return without performing any action at all.

User Commands

The USER commands provide useful replacements for BLOCK-READ and BLOCK-WRITE and allow you to perform soft reset and checksum functions. Not all of the USER commands have been fully emulated on RAMDrive. Those commands which are considered to be burst commands have not been included, and the USER commands normally used to perform machine language jumps into certain buffer locations have been disabled. All USER commands are sent via the DOS command channel (secondary address of 15).

U0 Utility Commands

The U0 command has a number of uses. Without any parameters, it resets the user vectors. When combined with specific parameters, U0 provides a way to send burst and utility commands. Here is the syntax:

```
PRINT#lf,"U0[+|-|>[ut]]"[+CHR$(d)]
```

where:
- lf = the logical file number used for the command channel
- ut = utility command character (if needed)
- d1 = burst utility data byte (if needed)
- d2 = second burst utility data byte (if needed)

Since most of these commands are covered in greater detail in the paragraph titled 'CHGUTL Utility' in the area describing burst commands, we will present only the general syntax of each command here. Assume that each the command begins with a 'PRINT#*lf*,' and that the *lf* should be replaced with an already-opened logical file number for the command channel.

"U0"	*Reset user vectors to default
"U0>"+CHR$(d)	Change device number (d = dev.#)
"U0>C"	Test RAM checksum
"U0>T"	*Test ROM checksum

*Denotes commands which are accepted but do not perform any function.

Note: These commands can be sent with the JiffyDOS wedge with the exception of the command used to change the device number. Commands which require the addition of a character string code to the command cannot be properly parsed by the JiffyDOS wedge.

JiffyDOS Examples:

```
@U0>C
@"U0>T"
@"U0",16
```

Reading Blocks with U1

The U1 command is used for reading specific blocks within a partition. U1 should be considered a direct access command, as is the BLOCK-READ command which U1 is normally used in place of. The U1 command will read an entire block into the buffer which is being used by the direct access channel. The syntax for the U1 command is:

```
PRINT#lf,"U1";ch;n;t;s
```

where: lf = the logical file number used for the command channel
ch = the channel number used for the direct access file
n = partition number (always 0)
t = the track from which you wish to read a block
s = the sector which you wish to read from

The channel number is the same as the secondary address used when opening the direct access file.

The partition number should always be 0 (zero). This is because direct access commands will always access the partition that was the current partition at the time the direct access channel was opened.

After using the U1 command to read a block, you may retrieve the data it stores in the buffer with the GET# command. You may position the pointer with the BUFFER-POINTER command to retrieve specific bytes before using the GET# command. You may also substitute the characters UA for U1. An example of this command can be found under the heading 'Reading and Writing Data with Direct Access' earlier in this section.

Writing Blocks with U2

The U2 command is used for writing specific blocks to a partition. U2 is considered to be a direct access command, as is the BLOCK-WRITE command which U2 is normally used in place of. The U2 command will write an entire block from the buffer which is being used by the direct access channel. The syntax for the U2 command is:

```
PRINT#lf,"U2";ch;n;t;s
```

where: lf = the logical file number used for the command channel
ch = the channel number used for the direct access file
n = partition number (always 0)
t = the track to which you wish to write a block
s = the sector which you wish to write to

The channel number is the same as the secondary address used when opening the direct access file.

The partition number should always be 0 (zero). This is because direct access commands will always access the partition that was the current partition at the time the direct access channel was opened.

Before using the U2 command, you should place the data you wish to write into the direct access channel's buffer. This may be accomplished by using the PRINT# command. You may place data in specific bytes by using the BUFFER-POINTER command before sending data with PRINT#. You may also substitute the characters UB for U2. An example of this command can be found under the heading 'Reading and Writing Data with Direct Access' earlier in this section.

User Jump Commands

The majority of the remaining USER commands perform jumps to certain routines in drive memory. Most drives support eight USER JUMP commands in all, but RAMDrive supports only two of these. The first six commands, U3 through U8 (alias UC through UH) were normally used to perform jumps to user installed machine language programs located in a drive. Since RAMDrive does not contain its own microprocessor, and because routines normally placed into a drive are usually device specific, we felt that these commands would not enhance the performance or compatibility of RAMDrive. The remaining two USER commands are used to reset the drive to various degrees, and these have been incorporated into RAMDrive. Here are their definitions:

COMMAND	ALIAS	DEFINITION
U9	UI	Warm reset (minimal effect on drive variables)
U:	UJ	Cold reset (does not change current partition)

The syntax for the preceding USER commands is:

```
PRINT#lf,"Ux"
```

where: lf = the logical file number used for the command channel
x = the character of the desired USER command

Burst Commands

Since RAMDrive does not connect directly to the serial bus, it is not possible to use most of the common burst commands found on the 1571 and 1581 disk drives. In most cases, burst commands are used to provide high speed data transfer. As RAMDrive already provides faster transfer rates than is possible with any serial device, burst commands are not necessary.

Special Loaders

The Commodore DOS Utility Loader and Autoboot Loader have not been implemented into the DOS on RAMDrive. If you require automatic execution of certain programs, use RAMDrive's capability to autoboot a specific file to accomplish this.

Job Queue Instructions

The job queue buffers are special locations used to pass commands and parameters to RAMDrive for the purpose of performing specific RAM disk access functions. The job queue can be accessed directly by the programmer if desired. Different areas are defined for job queues in RAMDrive depending on which partition type is currently being used. This was done so that RAMDrive could emulate the job queue locations of the 1541, 1571, and 1581. In all cases, the job codes and parameters are moved to the Native Mode job queue for actual execution. When writing job queue routines for RAMDrive, it is better to use the Native Mode job queue locations, since the routines will then operate no matter which type of partition is in use.

Job Queue Command Codes

Code	Name	Description
$80	READ	Reads logical block using track and sector parameters
$90	WRITE	Writes logical block using track and sector parameters
$92	DSK_IN_DRV	Not implemented (returns OK status)
$94	ACTIV_ON	Turns ACTIVITY LED on
$96	ACTIV_OFF	Turns ACTIVITY LED off
$98	ERR_ON	Turns ERROR LED on
$9A	ERR_OFF	Turns ERROR LED off
$A0	VERIFY	Not implemented (returns OK)
$B0	SEEK	Not implemented (returns OK)
$C0	BUMP	Not implemented (returns OK)
$D0	JUMP	Not implemented (returns OK)
$E0	EXEC	Not implemented (returns OK)
$F0	FRMT_PART	Not implemented (returns OK)

Job Queue Locations

Address Range	Description
$0000 - $0004	1541/1571 Emulation Mode Job Queue
$0006 - $000F	1541/1571 Emulation Mode Track & Sector
$0002 - $000A	1581 Emulation Mode Job Queue
$000B - $001C	1581 Emulation Mode Track & Sector
$0020 - $0027	Native Mode Job Queue
$0028 - $0037	Native Mode Track & Sector

Direct RAM Access

Direct access to any memory contained within RAMDrive may be performed through a set of routines and registers. The register arrangement is similar to the register arrangement used in Commodore REUs, though the base address for these registers differs. This should make it much easier to convert software originally intended for the 17xx series of REUs to utilize RAMDrive. Bear in mind that using this type of access to write data directly into the RAM is likely to corrupt data stored in RAM by RD DOS routines.

Direct RAM Access Routines

There are two direct ways of transferring or swapping memory contents between RAMDrive and the host computer. The first method uses registers in a manner which is similar to that used by Commodore REUs. The second method allows you to transfer a block of data directly between the computer and RAMDrive by using partition, track and sector information to specify the memory areas to be moved in RAMDrive.

REU Style Memory Moves

The main difference is that the Commodore REU uses an interrupt based scheme to initiate this type of transfer, while RAMDrive requires the controlling program to call an execute routine. RAMDrive also requires some other calls in order to make the RAMDrive hardware visible to the system, and to set up the REC Image Page registers. The following procedure outlines how to perform an REU type direct memory transfer:

1. Switch in RAMDrive Hardware ($E0A9 or $E0B1 shuts off interrupts)
2. Set REC Image Page ($FE03)
3. Set Registers ($DE00 - $DE0E)
4. Call execute ($FE06 or $FE1E)
5. Switch out RAMDrive Hardware ($FE0C will turn interrupts back on or $FE0F to exit without turning on interrupts)

Note: You may skip step 2 if you use $E0A9 for step 1, and you may skip step 5 if you use $FE1E for step 4.

Sector Addressed Memory Moves

The other method available for direct transfer of memory is intended for moving single 256 byte blocks at a time. These types of transfers use track and sector specifications to identify the RAMDrive memory to be transferred. This is similar to a block read or block write operation, but instead of having to read or write each individual block (as is the case with BLOCK-READ and BLOCK-WRITE), the entire block contents are automatically transferred between RAMDrive and the host computer's

memory. The following procedure outlines the use of this method of transfer:

1. Switch in RAMDrive Hardware ($E0A9 or $E0B1 shuts off interrupts)
2. Set REC Image Page ($FE03)
3. Set Registers ($DE20 - $DE25)
4. Call execute ($FE09 or $FE21)
5. Switch out RAMDrive Hardware ($FE0C will turn interrupts back on or $FE0F to exit without turning on interrupts)

Note: You may skip step 2 if you use $E0A9 for step 1, and you may skip step 5 if you use $FE21 for step 4.

Computer Memory Usage

It is important to remember that there are a few computer memory locations which are used by both of the methods outlined above. Four addresses in zero page are used, $00AE through $00B1. The contents of these four locations are saved when the Direct RAM Access execute routines are called, and are restored when the transfer is complete. This should have no effect upon the actual transfer being performed as long as you are not moving memory into zero page of the computer. If you are moving memory into zero page in the computer, then it will be necessary for you to place any required data into these four locations by using some alternate method.

Processor Stack Usage

The Direct RAM Access routines also use some of the processor stack area during transfers. Four bytes will be pushed on the stack when performing transfers in 64 mode, and five bytes are used for 128 mode transfers. All bytes placed on the stack by the transfer routines are removed from the stack at the end of the transfer. Again, this should have no detrimental effect on your routine unless you are low on stack memory or are attempting to transfer to stack memory. The latter is not recommended.

Direct RAM Access Jump Table

Normally, the RAMDrive Direct RAM Access jump vectors and REC Image Registers do not exist in memory. Only by performing a SYS or JSR to routines which enable these vectors and registers will you be able to perform direct access to all RAM within RAMDrive. The following is a brief description of the enable routines and associated vectors.

$E0A9	57513	EN_SET_REC	Routine to switch in RAMDrive Hardware, shut off interrupts, and set REC Image Page.
$E0B1	57521	RD_HW_EN	Routine to switch in RAMDrive Hardware and shuts off interrupts.

$FE03	65027	SET_REC_IMG	Routine to set REC Image Page
$FE06	(65030)	EXEC_REC_REU	Executes according to REU register settings
$FE09	(65033)	EXEC_REC_SEC	Executes according to sector move register settings
$FE0C	(65036)	RD_HW_DIS	Disables RAMDrive hardware and turns interrupts back on.
$FE0F	(65039)	RD_HW_DIS2	Disables RAMDrive hardware and leaves interrupts off.
$FE1E	(65054)	EXEC_REU_DIS	Executes according to REU register settings, disables RAMDrive hardware and turns interrupts back on.
$FE21	(65057)	EXEC_SEC_DIS	Executes according to sector register settings, disables RAMDrive hardware and turns interrupts back on.

Register Parameter Descriptions

The following descriptions should assist you in programming your own routines to use RAMDrive's Direct RAM Access routines. For some examples of using these routines and registers, you may wish to examine the RAM-TOOLS program, which makes use of both transfer methods.

$DE00 Status Register

Bits 0-3	Unused (always 0)
Bit 4	SIZE (always 1)
Bit 5	VERIFY ERROR (1=error / 0=OK)
Bit 6	TRANSFER COMPLETE
Bit 7	IRQ PENDING (always 0 - IRQ's not used in this scheme)

$DE01 Command Register

Bits 1 and 0	TRANSFER TYPE 00 Move memory from comp. to RAMDrive 01 Move memory from RAMDrive to comp. 10 Swap computer with RAMDrive 11 Verify computer RAM with RAMDrive
Bits 2 and 3	Reserved (no specific purpose)
Bit 4	$FF00 Decode (not implemented)
Bit 5	AutoLoad option - resets computer address, expansion address and transfer length to original values specified at beginning of transfer.

Bit 6 Reserved (No function)
Bit 7 Execute bit (No function)

$DE02-03 Computer Address Pointer

These two bytes form a pointer for the address in the computer which is to be used for memory transfers. Low byte goes in $DE02, high byte in $DE03.

$DE04-06 RAMDrive System Address Pointer

These three bytes form a pointer for the address in RAMDrive which is to be used for memory transfers. Low byte goes in $DE04, middle byte in $DE05, and the high byte in $DE06.

$DE07-08 Transfer Length

This byte is used to indicate how many bytes are to be transferred. Legal values are from 0-65535. Zero indicates that 65536 bytes are to be transferred.

$DE0A Address Control

This register is used to indicate how addresses should be incremented after the transfer occurs. Only bits 6 and 7 are significant.

Bits 6 and 7 Address Control
- 00 Increment both addresses (the computers address and RAMDrive address after you do your transfer)
- 01 RAMDrive address fixed
- 10 Computer address fixed
- 11 Both addresses fixed

New Registers in RAMDrive

$DE0E Transfer Error

Bit 7 of this register will be set to a value of 1 if an illegal block error occurs while the execute routine is operating. If no illegal block error was encountered, bit 7 will be cleared (set to 0). Errors also effect the carry flag, which is cleared when the RAMDrive block was valid, or set if it was invalid.

$DE10 Bank in 128 for Memory Transfer (128 mode)

Indicates which memory bank in the 128 is to be used for memory transfers. Bank numbers used for this location are the same as are used by BASIC 7.0 BANK command.

$DE20 JOB

This register is used to indicate the type of transfer desired. It is also used for reading back errors after the transfer is complete. The following is a list of acceptable job codes:

$80 READ
$90 WRITE
$A0 VERIFY
$B0 SWAP)

ERRORS: $02 VERIFY ERROR
$04 ILLEGAL BLOCK
$0F ILLEGAL PARTITION

$DE21 Track

This register is used to indicate the track number on RAMDrive where the block with which the transfer is to be performed is located.

$DE22 Sector

This register is used to indicate the sector number on RAMDrive where the block with which the transfer is to be performed is located.

$DE23-24 Computer Transfer Address

These two bytes form a pointer to the starting address in the computer which will be used for the block transfer.

$DE25 Partition in RAMDrive

This register indicates the partition number in RAMDrive where the block to be used for the transfer is located. The actual partition number should be used with two exceptions; a zero ($00) is used to indicate that the current partition should be used, and a hex value of $FF indicates that the transfer should occur with the system partition.

$DE10 Bank in 128 for Memory Transfer (128 mode)

Indicates which memory bank in the 128 is to be used for sector transfers. Bank numbers used for this location are the same as are used by BASIC 7.0 BANK command.

REC Image Page and other Registers		
Address		Definition
$DE00	56832	Status Register
$DE01	56833	Command Register
$DE02	56834	Computer Address (low byte)
$DE03	56835	Computer Address (high byte)
$DE04	56836	RAMDrive Address (low byte)
$DE05	56837	RAMDrive Address (middle byte)
$DE06	56838	RAMDrive Address (high byte)
$DE07	56839	Transfer Length (low byte)
$DE08	56840	Transfer Length (high byte)
$DE09	56841	Not used
$DE0A	56842	Address control register
$DE0E	56846	ERROR
$DE10	56848	Bank in 128 for transfer
$DE20	56834	Job
$DE21	56834	RAMDrive Track
$DE22	56834	RAMDrive Sector
$DE23	56834	Computer Address (low byte)
$DE24	56834	Computer Address (high byte)
$DE25	56834	RAMDrive Partition Number
$DE26	56834	Bank in 128 for sector transfer

Appendix A
Utilities

About the Utility Disks

A number of utility programs created by CMD are provided with RAMDrive. These programs are located on the floppy disk titled RAMDrive Utilities. Be sure to backup this disk immediately. The following is a list of the programs supplied with RAMDrive:

RAMDrive Utilities

RD-INSTALL	Installs RD DOS on RAMDrive
BBG-TOGGLE	Toggles BBG status
RAM-TOOLS	Partition and configuration utilities
FCOPY	File copier
MCOPY	Whole disk/partition copier
1541SUB	Subdirectory creation utility
1581SUB	Subdirectory creation utility
AUTOFILE EDITOR	Auto-start file configuration editor
AUTO-BOOT 128	Boot-block creation utility for the C128
DISK CRACKER HD	Track and sector editor and drive monitor
HARDWARE TEST	Checks system for RAMDrive compatibility
RAM TEST	Tests all RAM connected to the system
ZAP SYSTEM	Erases current system configurations parameters

Program Documentation

The various programs on this disk are in many cases discussed in other sections of this manual, while actual documentation is provided in this Appendix. Some programs are public domain utilities that have proven to be useful with the RAMDrive. These programs may not be fully documented here, but enough information has been included to allow those familiar with these types of utilities to begin using them. Some of the files located on the RAMDrive Utilities disk are machine language modules or data files which are used by the other programs, and require no separate documentation.

RD-INSTALL

RD-INSTALL installs into RAMDrive the Kernal overlays, which allow RAMDrive to capture serial bus commands, and it also installs RD DOS, the code that allows RAMDrive to emulate Commodore DOS compatible disk drives. RD-INSTALL will prompt the user for any required actions.

C128 computers must run RD-INSTALL in the C128 mode; because the program needs to detect the C128 version.

To run RD-INSTALL type:

```
LOAD"RD-INSTALL",8

RUN
```

Follow the prompts presented by the program. RD-INSTALL will cause the activity light on RAMDrive to pulse, then BBG will light up and the computer will execute a software reset.

BBG-TOGGLE

BBG-TOGGLE toggles the state of the BBG register and then performs a software reset. If BBG is off, then BBG-TOGGLE will turn BBG on. If BBG is on, BBG-TOGGLE will turn BBG off.

BBG must be on and RD DOS must be installed for RAMDrive to operate; see Section 3, BBG. BBG-TOGGLE is most useful for re-activating RAMDrive after a long reset; see Section 3, Reset Switch, Long Reset. RD-INSTALL will have to be re-run if RD DOS has been corrupted.

To run BBG-TOGGLE type:

```
LOAD"BBG-TOGGLE",8

RUN
```

RAM-TOOLS

RAM-TOOLS is used to create and delete partitions on RAMDrive, or to change RAMDrive's special configuration parameters. From 64 or 128 mode, load and run RAM-TOOLS. You will be presented with a menu containing several items. These items may be selected by pressing the appropriate number key shown next to the item on the main menu. A description of each of these items and their use is given below.

Default Device Number

RAMDrive's auto-configuration will automatically set the default device number to 16. You may change this default to any number between 8 and 29. To do so, select the CHANGE DEFAULT DEVICE NUMBER option from the main menu. The program will display the current RAMDrive default device number, and will ask if you wish to change this default. You may press the 'N" key to return to the main menu. If you press 'Y', then you will be prompted to select a new device number .Make your selection with the <+> and <-> keys and then press the <RETURN> key. The program will then ask if you wish to write the new configuration to RAMDrive. This change will remain in effect until you either change it again, or RAMDrive loses power entirely.

We recommend a default value of 9 if you have only one floppy disk on your system, or device number 10 if two floppy drives are being used (remember - you can always use the SWAP functions to switch RAMDrive's device number with devices #8 and #9). You should not set RAMDrive's default device number to a value which is normally used by another drive, unless you do not plan to use that drive and RAMDrive at the same time..

Default Partition Number

This option allows you to change which RAMDrive partition will be the active partition when your computer is powered up, or when the computer or RAMDrive are reset. Auto-configuration sets this default for partition number 1. The process for setting the default partition is similar to the process for setting the default device number. The program will check to see if the partition you select is legal.

Select the CHANGE DEFAULT PARTITION NUMBER option from the menu. The program displays the current default partition number, and will then prompt you to select a new default. Make your selection with the <+> and <-> keys and press the <RETURN> key.

View Partition Table

This option shows the current status of all partitions. The <+> and <-> keys will step you through the pages in the display, and <RETURN> allows you to exit back to the main menu.

Create a New Partition

This option allows you to add new partitions to the system. The program will automatically default to the next available (unused) partition. This is usually the best choice as it will keep your partitions in order. However, if you wish, you may select any partition not yet in use. Use <+> and <-> to change the partition number, and press the <RETURN> key to accept.

You must then select the partition type. Again, use <+> and <-> to view the available choices, and <RETURN> to accept.

If you have not selected an Emulation mode partition, you will need to specify the partition size. The size is adjustable in increments of 256 blocks using <+> and <-> to select, and <RETURN> to accept.

The last step is to enter a name for the partition (16 characters maximum). Even though the usable characters have been limited in this program, you can later rename the partition with the RD DOS Rename Partition command if necessary. After entering the partition name you will be given the option to create the new partition on RAMDrive or to abort the process.

Delete an Old Partition

Upon choosing this option you will be shown the first partition on RAMDrive. To select the partition that you wish to delete, use the <+> and <-> keys, and then press <RETURN> to accept. After the partition has been selected, you will be given an opportunity to abort the process before the partition is deleted.

Deleting a partition can take from a few seconds to several minutes depending on where the partition to be deleted is located on RAMDrive. Partitions located above (at a higher address than) the partition being deleted are moved down to fill the void left by the deleted partition. This is done in order to avoid the fragmentation of storage space. If you are going to delete a number of partitions, start with the last one created, and then continue in reverse order of creation. This will save a lot of time. If you want to create all new partitions from scratch, it may be faster to remove power from RAMDrive, allowing it to create only one or two partitions with .

WARNING: Deleting partitions will destroy all data within the partitions you are deleting. Do not delete partitions without first backing up any data in those partitions which you wish to keep.

Quit

Allows you to exit from RAM-TOOLS. Remember to press the RESET switch on RAMDrive before trying to use it again.

Do's and Don'ts

Do:

- plan ahead. Set up your partitions logically.
- backup important data before doing any system configuration. Chances for data loss are much greater when performing these functions.

Don't:

- press RESET or any other switches on RAMDrive while a file is open or there is any RAMDrive activity.
- make a regular habit of partitioning. This is something you should do once or twice, and edit only when your needs change.

FCOPY

This program was created by CMD to fill the need for a file copier capable of copying any type of file between any two drives or between any two partitions on our devices. In addition, FCOPY supports Native Mode subdirectories, 1581 sub-partitions, and 17xx REU's running under Commodore RAMDOS. The use of this program is mostly self-explanatory, but we have included a breakdown of the functions here in order to provide more detail where necessary. This program may also be used to remove files, view directories, and send disk commands.

Set Source Device (F1)

This option selects the disk drive that you wish to copy files *from*. The device number and type of device will be shown on the display. If the drive type is not recognized by the program, question marks will be shown instead of the actual type. In the case of unrecognized third party disk drives, the program will work regardless unless the DOS in the particular drive happens to be highly incompatible with standard Commodore-style DOS commands and file handling procedures.

Set Target Device (F5)

This option selects the disk drive that you wish to copy files *to*. The device number and type of device will be shown on the display.

Set Source Partition (F3)

If the source device is a CMD HD, CMD RD, or CMD RL, this option allows you to select the partition from which files are to be copied. The partition number, name and type will be displayed in the source area.

Set Target Partition (F7)

If the target device is a CMD HD, CMD RD, or CMD RL, this will allow you to set the partition to which files are to be copied. The partition number, name and type will be displayed in the target area.

Set Source Path (S)

If the source device is a CMD HD, CMD RD, or CMD RL, and the partition type is Native or 1581 Emulation, or if the source device is an

actual 1581, a path for subdirectories or sub-partitions may be entered. Each subdirectory or sub-partition name must be separated by a slash (/).

Set Target Path (T)

Same function as Set Source Path, except that the path is intended for the Target device.

Source/Target Directory (A/B)

Allows you to view the directory of the source or target disk. The program will ask for a search pattern so that you may view files which match a given name and/or filetype. For directories read from a CMD HD, CMD RD, or CMD RL, the pattern will also allow the selection of files by time and date.

Select Files (F)

This option reads the source directory into the selection buffer after you have entered a search pattern. File information is stored in a dynamic method, so if necessary, you will be able to view from 700 to well over a thousand files. If your directory contains more files than can be viewed, you will need to limit the selection by using a pattern which will match fewer files. Once you have entered the file selection mode, you may select or de-select files by pressing the RETURN key while the arrow is pointing to the file. An asterisk (*) indicates which files have been selected. Pressing the 'T' key allows you to toggle all selections ('T' will select all unselected files, and de-select all selected files). When you have finished, press the back-arrow key (←) to return to the main menu.

Reselect Files (R)

You may, at any time after selecting files, return to the file selection mode without reading the directory. This allows you to change your selections before or after copying files.

Send Disk Command (@)

This option allows you to send a disk command to either the source or the target drive. Important: disk commands do not follow the source or target path, but are instead sent to the current directory. Therefore, it is wise to make sure you are currently in the correct partition, sub-partition, or subdirectory before sending a command intended for a certain area. When sending commands to RAMDrive, you may include a partition number and path in the disk command.

Begin Copying (C)

This option starts the copying process. Only the files you have selected using the 'F' and/or 'R' options will be copied. When copying is complete,

an option allows you to copy the same files to another disk (if your target drive is a floppy disk drive).

Begin Scratching (#)

This option starts scratching the files you selected using the 'F' and/or 'R' options. Please note: files can only be scratched from the source disk! You will be asked "ARE YOU SURE (Y/N)?" before the scratching operation begins.

Exit Program (←)

Allows you to exit from program. If you wish to use the program again, it must be re-loaded.

MCOPY

This program is of the type commonly referred to as a 'whole disk copier'. It can copy an entire disk between two floppy drives of the same type, a floppy drive and a similar CMD RD partition, two similar partition types on the same RAMDrive or two similar partitions on separate CMD hard drives. The program will support all Commodore floppy drives, as well as any fully compatible third party drives.

MCOPY is also useful for backing up partitions to a floppy disk or to other partitions on RAMDrive. It is also possible to copy Native Mode partitions to other Native Mode partitions of different sizes. This can be useful when you decide that you need a larger partition to hold data. Although it is also possible to copy from a larger partition to a smaller one, it is possible that some data will be lost in this process.

MCOPY is self-documenting. A help menu is always on screen and lists the available options.

1541SUB and 1581SUB

These utilities are nearly identical in function and are used to create Native Mode subdirectories which emulate the directories of 1541 and 1581 drives. Using these types of subdirectories may allow the use of programs which will not normally work in a Native Mode partition.

> WARNING: Use these utilities on an empty Native Mode partition only. Any data stored in the partition will be lost.

1541SUB creates a subdirectory which begins at the same location as the directory on a 1541 (track 18, sector 1). A header block for this directory is also created and placed where the header block on a 1541 is located (track 18, sector 0).

1581SUB creates a subdirectory which begins at the same location as the directory on a 1581 (track 40, sector 3). A header block for this directory is also created and placed where the header block on a 1581 is located (track 40, sector 0). Due to the way Native Mode subdirectories work, 1581SUB must create two subdirectories to provide the proper headers and directory space. This program has been tested for use with Superbase and Superbase 128 and allows these programs to access larger storage areas.

To use either of these programs, LOAD and RUN the one which you wish to use. The program will ask for the device number of RAMDrive as well as the partition number you wish to create the subdirectory in. Do not use either of these programs on a partition which contains useful data, as the partition will be formatted by the utility. You must also make sure that the partition you select for use has enough tracks to support the subdirectory (minimum 18 tracks for 1541SUB, 40 tracks for 1581SUB).

AUTOFILE EDITOR

This utility allow you to create and edit special configuration data stored on RAMDrive. By using this program, you can configure RAMDrive to automatically load any file you wish from any drive on your system each time the computer is turned on or reset. Two separate areas exist for this purpose - one for a C64 or C128 in 64 mode, and another area for a C128 in 128 mode. This allows 128 users to have separate boot files for each mode on their machines. There are a total of six parameters which may be set for each of these areas. These are:

1. Enable / Disable Autofile loading
2. BASIC or Machine Language file
3. SYS Address if program is Machine Language
4. Device Number of drive where the file is located
5. Memory Bank for load and SYS if file is Machine Language (128 only)
6. Path and Filename of file to be loaded

The program is menu driven, and will only present the options available according to your current mode and settings. It is wise to leave the autofile option disabled until you have properly set the other parameters.

AUTO-BOOT 128

This Public Domain utility has been included to allow you to create boot sectors for use in 128 mode. A boot sector may be created in any Emulation Mode or Native Mode partition. Native Mode partitions automatically protect the boot sector keeping it allocated at all times. Therefore, when using this program on a Native Mode partition, you will be told that the sector is already being used. If you continue with the process at this point, the program will go ahead and create the boot sector.

Before running this program, use the Change Partition command to make the partition that you wish to place the boot sector in the current partition. You may then specify drive 0 when asked for this information within the program.

DISK CRACKER HD

This utility is a modification of a program released to the public domain by its author, Mike Henry. DISK CRACKER HD is a disk editor and monitor intended for use only by those familiar with programs of this type. We will not make any attempt to document this software, since it should only be used by those familiar with the data storage methods used on Commodore compatible disk drives, and these individuals are usually well versed in the use of this type of software.

HARDWARE TEST

This program is used to test the compatibility of your particular hardware with the standard RAMDrive unit. It is entirely possible that RAMDrive will not operate with some computers due to timing inconsistencies caused by wide variations in the manufacture of Commodore computers. If you notice inconsistent operation with RAMDrive (i.e., computer lockup, reset or bombing out of programs) which are not attributable to normal software incompatibility, you should run this test program. Follow the directions given in the program itself after loading and running it. You should allow this test to operate for several hours, starting after your computer has been turned off for several hours. This will allow the program to test your computer at all operating extremes. If this test fails, contact CMD for information on how to adapt your system for use with RAMDrive.

RAM TEST

This program will test all RAM attached to RAMDrive. Whenever you suspect problems with the RAM, load and run this program. Follow the on-screen prompts. Please note that this test is a destructive test, and you will have to reconfigure your system after running it.

ZAP SYSTEM

This program will allow you to completely destroy all configuration and partitioning information stored in RAMDrive. This will destroy any data you may have stored on RAMDrive, so use it only if you do not wish to retain that data on RAMDrive.

Appendix B
Error Codes

The error codes used by RAMDrive have been arranged to be compatible with the codes used on Commodore floppy disk drive units. Some errors have been eliminated on RAMDrive since certain conditions that exist in floppy drives do not occur in the same manner on a RAMDrive. Whenever errors are encountered on your RAMDrive, these codes should help in localizing the problem.

Errors are returned over the command channel in the following format:

```
ec,estring,tv,sv
```

where:
- ec = a two-digit error number
- estring = an ASCII string describing the error
- tv = the track variable (the logical track where the error took place)
- sv = the sector variable (the logical sector where the error took place)

Note: Some errors define special meanings for the track and sector variables and are described below when necessary.

Command Channel Error Codes

00 **OK** (not an error)
This code is present when no other error condition exists.

01 **FILES SCRATCHED** (not an error)
Occurs after using the DOS or BASIC scratch commands. The number of files scratched will be indicated in the track variable, and the sector variable will contain a zero.

02 **PARTITION SELECTED** (not an error)
Occurs after switching partitions with the RD DOS 'CP' command, and after changing 1581 sub-partitions with the '/' command. After using the 'CP' command, the track variable contains the partition number of the newly selected partition, while the sector variable contains zero. After the '/' command is issued, the track variable contains the number of the first track in the 1581 sub-partition, while the sector variable contains the number of the last track in the 1581 sub-partition.

26 **WRITE PROTECT ON**
Indicates that an attempt was made to write to a RAMDrive partition while that partition was write protected.

30 **SYNTAX ERROR** (general)
Indicates that the RD DOS command interpreter was unable to identify the last command sent via the command channel. This is usually caused by incorrect characters being present in the disk command.

31 **SYNTAX ERROR** (unrecognized command)
Usually indicates that the first character of the last command string was not recognized as part of a legal DOS command.

32 **SYNTAX ERROR** (command string too long)
Occurs when a disk command string contains over 254 characters. Due to the large size of the command channel input buffer in RAMDrive, this error should rarely be encountered.

33 **SYNTAX ERROR** (illegal file name)
Usually indicates that an attempt was made to use wildcards or pattern matching within a file name or disk command that does not accept wildcards.

34 **SYNTAX ERROR** (missing file name)
The last disk command failed due to a missing file name or the file name should have been preceded by a colon (:).

48 **ILLEGAL JOB** (in job queue)
The last job code placed into the job queue was not legal.

50 **RECORD NOT PRESENT**
The last attempt to access a relative (REL) file record specified a record number which does not yet exist. This condition will result even when attempting to create or expand a relative file, and under those conditions should be ignored.

51 **OVERFLOW IN RECORD**
The last attempted write to a relative record contained more data than could be stored in the record. When this occurs, as much data as can fit is stored into the record.

52 **FILE TOO LARGE**
The last attempt to access a relative record would have exceeded the amount of storage space remaining in the partition specified. Under this condition, no additional records were added to the file, and if this was an attempt to create a new relative file, the file was not created.

60 **WRITE FILE OPEN**
The last attempt to open a file was made to a file which was already open for writing.

61 **FILE NOT OPEN**
Indicates that the last attempt to access data was made to a file which was not properly opened. Under some circumstances, this error is not generated and the attempt to access the file is simply ignored.

62 **FILE NOT FOUND**
During the last attempt to open a file, the DOS could not find the file specified by the path and filename given. This error will also occur if the filetype of the file does not match the allowed filetypes.

63 **FILE EXISTS**
While attempting to open a new file, another file with the same name was found in the specified partition or sub-directory.

64 **FILE TYPE MISMATCH**
The last disk operation specified a file which did not match the filetypes allowed for that operation.

65 **NO BLOCK**
Indicates that an attempt was made to allocate a block which was already allocated using the BLOCK-ALLOCATE (B-A) command. The track and sector variables will contain the next available block when this condition occurs. If the track variable contains a zero, there are no blocks available.

66 **ILLEGAL BLOCK**
Occurs when an attempt is made to access a track or sector which does not exist. This may occur if one of the track and sector links within a file have become corrupted.

67 **ILLEGAL BLOCK**
Usually caused by a corrupt disk. This error shows up when the block parameters for a partition do not exist.

70 **NO CHANNEL**
This indicates that the specified channel within a disk command is already in use, or that all buffers in the drive are currently in use. This may be an indication that too many files are currently opened.

71 **DIRECTORY ERROR**
Indicates that the BAM (Block Availability Map) on the diskette is invalid. To correct the problem, Validate the disk.

DIRECTORY FILE ENTRY FORMAT ALL PARTITION TYPES AND NATIVE MODE SUBDIRECTORIES		
BYTE	**VALUE**	**DESCRIPTION**
0		File type:
	0	DEL (Deleted)
	1	SEQ (Sequential
	2	PRG (Program)
	3	USR (User)
	4	REL (Relative)
	5	CBM (1581 style sub-partition)
	6	DIR (Native Mode subdirectory)
		Note: Filetypes will be OR'ed with $80 when the file has been properly closed. Filetypes will be OR'ed with $C0 if the file is locked.
1		Track pointer to first data block (or header block if filetype is DIR)
2		Sector pointer to first data block (or header block if filetype is DIR)
3 - 18		Filename padded with shifted spaces ($A0)
19		Pointer to starting track of side sector or super side sector if filetype is REL
20		Pointer to starting sector of side sector or super side sector if filetype is REL
21		Record length if filetype is REL
22	0	Reserved
23		Year file was created (last two digits)
24		Month file was created
25		Day file was created
26		Hour file was created
27		Minute file was created
28		Number of blocks used by file (low byte)
29		Number of blocks used by file (high byte)

Figure C2

1541 and 1571 Emulation Mode Partitions

SECTORS PER TRACK (1541 EMULATION MODES)		
TRACK RANGE	**SECTORS AVAILABLE**	**TOTAL**
1 through 17	0 through 20	21
18 through 24	0 through 18	19
25 through 30	0 through 17	18
31 through 35	0 through 16	17

Figure C3

SECTORS PER TRACK (1571 EMULATION MODE)		
TRACK RANGE	**SECTORS AVAILABLE**	**TOTAL**
1 through 17	0 through 20	21
18 through 24	0 through 18	19
25 through 30	0 through 17	18
31 through 35	0 through 16	17
36 through 52	0 through 20	21
53 through 59	0 through 18	19
60 through 65	0 through 17	18
66 through 70	0 through 16	17

Figure C4

HEADER & BAM (1541 & 1571 EMULATION MODES) TRACK 18 SECTOR 0		
BYTE	**VALUE**	**DESCRIPTION**
0	18	Track pointer to first directory block
1	1	Sector pointer to first directory block
2	65	ASCII 'A' for format type
3	0	1541 Emulation Mode
	128	1571 Emulation Mode
4 - 143		BAM (Block Availability Map)
144 - 161		Disk name padded with shifted spaces
162 - 163		Disk ID
164	160	Shifted Space for separator
165	50	ASCII '2' for DOS version
166	65	ASCII 'A' for format type
167 - 170	160	Shifted spaces for separators
171 - 220	0	Null bytes - reserved
221 - 255	0	1541 Emulation mode - Null bytes - reserved 1571 Emulation mode - Number of sectors available for tracks 36 through 70 - one byte per track (part of 1571 (side 2) BAM)

Figure C5

BAM for 1571 (side 2) EMULATION MODE TRACK 53 SECTOR 0		
BYTE	**VALUE**	**DESCRIPTION**
0 - 104		BAM for tracks 36 through 70 (3 bytes per track)
105 - 255	0	Null bytes - reserved

Figure C6

BAM ENTRY FORMAT 1541 & 1571 (side 1) EMULATION MODES Format of bytes 4-143 in Track 18 Sector 0 (Figure C5) 4 bytes per track: bytes 4-7 cover track 1, bytes 8-11 cover track 2, ...	
BYTE	**DESCRIPTION**
0	Number of sectors available on track
1	Block Availability for sectors 0 - 7
2	Block Availability for sectors 8 - 15
3	Block Availability for sectors 16 - 23
Notes: The lowest bit (LSB) in each byte (bytes 1 through 3) indicates the status of the lowest sector covered by that byte. A binary value of 1 indicates that the sector is available, while a value of 0 indicates that the sector is allocated.	

Figure C7

BAM ENTRY FORMAT 1571 (side 2) EMULATION MODE Format of bytes 0 - 104 on Track 53 Sector 0 (Figure C6) 3 bytes per track: bytes 0-2 cover track 36, bytes 3-5 cover track 37, ...	
BYTE	**DESCRIPTION**
0	Block Availability for sectors 0 - 7
1	Block Availability for sectors 8 - 15
2	Block Availability for sectors 16 - 23
Notes: The lowest bit (LSB) in each byte (bytes 0 through 2) is used to indicate the status of the lowest sector covered by that byte. A binary value of 1 indicates that the sector is available, while a value of 0 indicates that the sector is allocated. The associated byte for the number of sectors available on each track is stored in bytes 221 through 225 of track 18 sector 0 (see Figure C5).	

Figure C8

1581 Emulation Made Partitions

SECTORS PER TRACK		
TRACK RANGE	**SECTORS AVAILABLE**	**TOTAL**
1 through 80	0 through 39	40

Figure C9

DIRECTORY HEADER TRACK 40 SECTOR 0		
BYTE	**VALUE**	**DESCRIPTION**
0	40	Track pointer to first directory block
1	3	Sector pointer to first directory block
2	68	ASCII 'D' for format type
3	0	Reserved
4 - 21		Disk name padded with shifted spaces
22 - 23		Disk ID
24	160	Shifted Space for separator
25	51	ASCII '3' for DOS version
26	68	ASCII 'D' for format type
27 - 28	160	Shifted spaces for separators
29 - 255	0	Null bytes - reserved

Figure C10

BAM BLOCK1 TRACK 40 SECTOR 1		
BYTE	**VALUE**	**DESCRIPTION**
0	40	Track pointer to next BAM block
1	2	Sector pointer to next BAM block
2	68	ASCII 'D' for DOS version
3	187	Complement of version number
4 - 5		Disk ID
6	192	Not used in RAMDrive - set to 1581 default value
7	0	Flag for Auto Loader file
8 - 15	0	Reserved
16 - 255		BAM for tracks 1 through 40 (6 bytes per track)

Figure C11

BAM 2 TRACK 40 SECTOR 2		
BYTE	**VALUE**	**DESCRIPTION**
0	0	Indicates last sector for BAM
1	255	Indicates all bytes in sector used
2	68	ASCII 'D' for DOS version (copy)
3	187	Complement of version number (copy)
4 - 5		Disk ID (copy)
6	192	Not used in RAMDrive - set at 1581 default value
7	0	Flag for Auto Loader file (copy)
8 - 15	0	Reserved
16 - 255		BAM for tracks 41 through 80 (6 bytes per track)

Figure C12

BAM ENTRY FORMAT Format of bytes 16 - 255 in Track 40 Sectors 1 and 2 (Figures C11 & C12)	
BYTE	**DESCRIPTION**
0	Number of sectors available on track
1	Block Availability for sectors 0 - 7
2	Block Availability for sectors 8 - 15
3	Block Availability for sectors 16 - 23
4	Block Availability for sectors 24 - 31
5	Block Availability for sectors 32 - 39
Notes: The lowest bit (LSB) in each byte (bytes 1 through 5) indicates the status of the lowest sector covered by that byte. A binary value of 1 indicates that the sector is available, while a value of 0 indicates that the sector is allocated.	

Figure C13

Native Made Partitions

SECTORS PER TRACK		
TRACK RANGE	**SECTORS AVAILABLE**	**TOTAL**
1 through 255	0 through 255	256

Figure C14

ROOT DIRECTORY AND SUBDIRECTORY HEADER TRACK 1 SECTOR 1 FOR ROOT DIRECTORY VARIES FOR SUBDIRECTORIES		
BYTE	**VALUE**	**DESCRIPTION**
0		Track pointer to first directory block
1		Sector pointer to first directory block
2	72	ASCII 'H' for format type,
3	0	Reserved
4 - 21		Disk name padded with shifted spaces
22 - 23		Disk ID
24	160	Shifted space for separator
25	49	ASCII '1' for DOS version
26	72	ASCII 'H' for format type
27 - 28	160	Shifted spaces for separators
29 - 31	0	Reserved
32	1	Pointer to ROOT header track
33	1	Pointer to ROOT header sector
34		Track pointer to DIR PARENT header
35		Sector pointer to DIR PARENT header
36		Track pointer to DIR entry in PARENT directory
37		Sector pointer to DIR entry in PARENT directory
38		Index to starting byte of DIR entry in PARENT directory
39 - 255	0	Null bytes - reserved

Figure C15

NATIVE MODE BAM (1st BAM block) TRACK 1 SECTOR 2		
BYTE	**VALUE**	**DESCRIPTION**
0	0	Reserved
1	0	Reserved
2	72	ASCII 'H' for format type
3	183	Complement of format type
4 - 5		Disk ID
6	192	Not used in RAMDrive - set at 1581 default value
7	0	Flag for Auto Loader file
8		Track number of last available track in partition
9 - 31	0	Reserved
32 - 255	0	BAM for tracks 1 through 7 (32 bytes per track)

Figure C16

NATIVE MODE BAM (blocks 2-32) TRACK 1 SECTORS 3 - 28		
TRACK	**SECTOR**	**DESCRIPTION**
1	3	BAM for tracks 8 through 15 (32 bytes per track)
1	4	BAM for tracks 16 through 23 (32 bytes per track)
1	5	BAM for tracks 24 through 31 (32 bytes per track)
1	6	BAM for tracks 32 through 39 (32 bytes per track)
1	7	BAM for tracks 40 through 47 (32 bytes per track)
1	8	BAM for tracks 48 through 55 (32 bytes per track)
1	9	BAM for tracks 56 through 63 (32 bytes per track)
1	10	BAM for tracks 64 through 71 (32 bytes per track)
1	11	BAM for tracks 72 through 79 (32 bytes per track)
1	12	BAM for tracks 80 through 87 (32 bytes per track)
1	13	BAM for tracks 88 through 95 (32 bytes per track)
1	14	BAM for tracks 96 through 103 (32 bytes per track)
1	15	BAM for tracks 104 through 111 (32 bytes per track)
1	16	BAM for tracks 112 through 119 (32 bytes per track)
1	17	BAM for tracks 120 through 127 (32 bytes per track)
1	18	BAM for tracks 128 through 135 (32 bytes per track)
1	19	BAM for tracks 136 through 143 (32 bytes per track)
1	20	BAM for tracks 144 through 151 (32 bytes per track)
1	21	BAM for tracks 152 through 159 (32 bytes per track)
1	22	BAM for tracks 160 through 167 (32 bytes per track)
1	23	BAM for tracks 168 through 175 (32 bytes per track)
1	24	BAM for tracks 176 through 183 (32 bytes per track)
1	25	BAM for tracks 184 through 191 (32 bytes per track)
1	26	BAM for tracks 192 through 199 (32 bytes per track)
1	27	BAM for tracks 200 through 207 (32 bytes per track)
1	28	BAM for tracks 208 through 215 (32 bytes per track)
1	29	BAM for tracks 216 through 223 (32 bytes per track)
1	30	BAM for tracks 224 through 231 (32 bytes per track)
1	31	BAM for tracks 232 through 239 (32 bytes per track)
1	32	BAM for tracks 240 through 247 (32 bytes per track)
1	33	BAM for tracks 248 through 255 (32 bytes per track)

Figure C17

NATIVE MODE BAM ENTRY FORMAT Format of bytes 32 - 255 in Track 1 Sector 2 and bytes 0 - 255 in Track 1 Sectors 3 - 33 (Figures C11 & C12)	
BYTE	**DESCRIPTION**
0	Block Availability for sectors 0 - 7
1	Block Availability for sectors 8 - 15
2	Block Availability for sectors 16 - 23
3	Block Availability for sectors 24 - 31
4	Block Availability for sectors 32 - 39
5	Block Availability for sectors 40 - 47
6	Block Availability for sectors 48 - 55
7	Block Availability for sectors 56 - 63
8	Block Availability for sectors 64 - 71
9	Block Availability for sectors 72 - 79
10	Block Availability for sectors 80 - 87
11	Block Availability for sectors 88 - 95
12	Block Availability for sectors 96 - 103
13	Block Availability for sectors 104 - 111
14	Block Availability for sectors 112 - 119
15	Block Availability for sectors 120 - 127
16	Block Availability for sectors 128 - 135
17	Block Availability for sectors 136 - 143
18	Block Availability for sectors 144 - 151
19	Block Availability for sectors 152 - 159
20	Block Availability for sectors 160 - 167
21	Block Availability for sectors 168 - 175
22	Block Availability for sectors 176 - 183
23	Block Availability for sectors 184 - 191
24	Block Availability for sectors 192 - 199
25	Block Availability for sectors 200 - 207
26	Block Availability for sectors 208 - 215
27	Block Availability for sectors 216 - 223
28	Block Availability for sectors 224 - 231
29	Block Availability for sectors 232 - 239
30	Block Availability for sectors 240 - 247
31	Block Availability for sectors 248 - 255
Notes: The lowest bit (LSB) in each byte (bytes 0 through 31) indicates the status of the highest sector covered by that byte. A binary value of 1 indicates that the sector is available, while a value of 0 indicates that the sector is allocated.	

Figure C18

File Formats

PROGRAM FILE FORMAT	
BYTE	**DESCRIPTION**
0	Pointer to track of next file block (contains a zero if current block is last block in file).
1	Pointer to sector of next file block (contains pointer to last byte used if current block is last block in file).
2 - 255	Program data (bytes 2 and 3 contain load address of program in low byte-high byte format if current block is first block in file).

Figure C19

SEQUENTIAL FILE FORMAT	
BYTE	**DESCRIPTION**
0	Pointer to track of next file block (contains a zero if current block is last block in file).
1	Pointer to sector of next file block (contains pointer to last byte used if current block is last block in file).
2 - 255	Data bytes .

Figure C20

RELATIVE FILE DATA BLOCK FORMAT	
BYTE	**DESCRIPTION**
0	Pointer to track of next data file block (contains a zero if current block is last data block in file).
1	Pointer to sector of next data file block (contains pointer to last byte used if current block is last data block in file).
2 - 255	Data bytes .Empty records will begin with a $FF in the first byte of the record, the remaining bytes will contain $00 bytes. Partially filled records will also contain $00 bytes in the unused portion of the record.

Figure C21

RELATIVE FILE SUPER SIDE SECTOR BLOCK FORMAT	
BYTE	**DESCRIPTION**
0	Pointer to track of first side sector in first group (group 0).
1	Pointer to sector of first side sector in first group (group 0).
2	Super side sector identification byte ($FE)
3 - 254	Track and sector pointers to first side sector of 126 groups (groups 0 through 125, two bytes per pointer). Unused group pointers contain $00 bytes.

Figure C22

RELATIVE FILE SIDE SECTOR BLOCK FORMAT	
BYTE	**DESCRIPTION**
0	Pointer to track of next side sector in this group (contains a zero if current block is last side sector block in use).
1	Pointer to sector of next side sector in this group (contains pointer to last byte used if current block is last side sector block in use).
2	Side sector number (0 - 5)
3	Record length of associated relative file
4	Pointer to track of first side sector in this group (number 0).
5	Pointer to sector of first side sector in this group (number 0).
6 - 7	Pointer to track and sector of second side sector in this group (number 1).
8 - 9	Pointer to track and sector of third side sector in this group (number 2).
10 - 11	Pointer to track and sector of fourth side sector in this group (number 3).
12 - 13	Pointer to track and sector of fifth side sector in this group (number 4).
14 - 15	Pointer to track and sector of sixth side sector in this group (number 5).
16 - 255	Track and sector pointers to 120 data blocks (two bytes per pointer). Unused data block pointers contain $00 bytes.

Figure C23

Appendix D
RAMDrive Memory Map

RAMDrive Partitionable RAM

RAMDrive actually contains two separate memory maps. One is used by RD DOS for keeping track of important system variables, as well as for handling the emulation of many disk drive functions and memory locations (such as the Job Queue buffer). The second map contains all the partitionable RAM. A map of this area is given below.

The SYSTEM partition is always located at the very top of any available memory.

There is almost always some portion of the memory which cannot be partitioned.

Memory which is partitioned starts at the very bottom of available memory (memory address $0000). Bear in mind that the memory map used for memory is an alternate map which only contains partitionable RAM. Due to this method of handling the RAM, performing a MEMORY-READ to location $0000 of RAMDrive looks at a different area (DOS memory) than performing a Direct RAM Access read of location $0000 (which would look into the partitionable RAM).

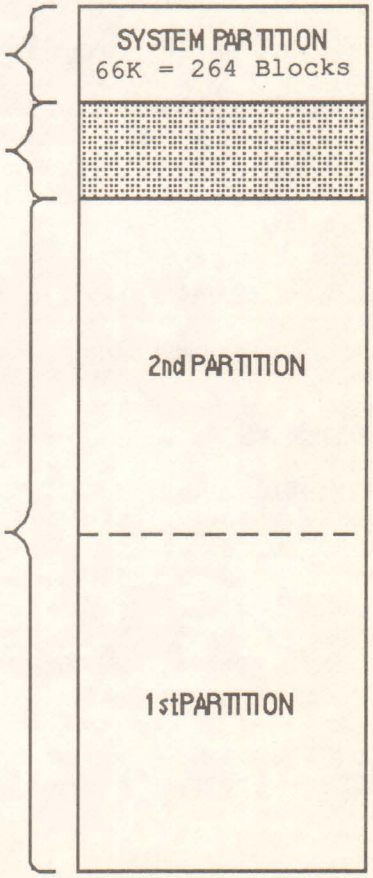


Figure D1

How Partitions are Allocated

Partitions are created beginning at the very bottom of partitionable memory. Partition numbers do not always correlate with a partitions actual placement. Instead, the order in which partitions are created determines where the partitions will be located in partitionable memory. For example, if you create partition number 1, then partition number 5, and finally partition number 2, then partition 1 will be the lowest in memory, partition 2 will be the highest in memory, with partition 5 existing between the two.

Deleting partitions will change this arrangement to some degree. Using the example just given, if you delete partition 5, then partition 2 will be moved so that it now begins where partition 5 used to be. This has the effect of filling up gaps in the partitionable memory, leaving you the maximum amount of RAM available for more partitions.

Since partitions do have certain size restrictions, there will almost always be some amount of RAM which will not be partitionable. Careful planning of your partitions can reduce this to a minimum. When planning partitions, it is usually wise to keep any Foreign (Direct Access) partitions at the very bottom of the memory map, so create these first. When figuring partition sizes it may help to know that there are 4096 Commodore blocks in 1 Megabyte.

Useful RAMDrive Memory Locations

The following chart gives information on a number of useful memory locations within the RAMDrive DOS area. Bear in mind that this area is separate from the area described above. Do not confuse the DOS RAM area with the SYSTEM partition - these are separate entities.

Address Range	Description of Contents
$0000 - $0004	1541/1571 Emulation Mode Job Queue
$0006 - $000F	1541/1571 Job Queue Track & Sector Variables
$0002 - $000A	1581 Emulation Mode Job Queue
$000B - $001C	1581 Job Queue Track & Sector Variables
$0020 - $0027	Native Mode Job Queue
$0028 - $002F	Native Mode Job Queue Track & Sector Variables
$0030 - $00FF	Emulated Zero Page Variables
$0100 - $01FF	Variables
$0200 - $027F	Input Command Buffer
$0280 - $02FF	Variables

Appendix E
Foreign REU Support

Commodore 17xx Series REU

RAMDrive is capable of working in parallel with a 17xx series REU. The two devices can co-operate on a cartridge port expander that connects each device to all of the signals simultaneously. Expanders that activate only one port at a time require modifications that will allow all lines to reach each device.

WARNING: Always backup your RAMDrive partitions and files before adding or removing an REU to a RAMDrive hardware configuration.

Once configured the RAMDrive will retain a map of the hardware which includes the REU. If the 17xx REU is physically added or removed, then the auto-configure process will adjust RAMDrive's hardware table, the memory will be re-partitioned, and reformatted. Loss of files and data will result. Therefore, backup the data before proceeding with hardware changes involving a 17xx REU.

RD DOS will auto-configure the RAMDrive and the REU as separate partitions. The entire 17xx REU becomes native mode partition 1. RAMDrive will have the system area and native mode partition 2.

Restrictions will apply to this hardware configuration; since the RAMDrive is battery backed and the REU is not. It is important that these two devices do not share a partition; since loss of power will leave RAMDrive with a partially valid partition. Loss of power will require the 17xx REU's partition(s) to be NEWed again, since the partition table in RAMDrive expects validly formatted partition(s) in the REU.

Programs that use the 17xx REU directly will access the device without RD DOS intercepting the command. These programs will destroy any formatted partitions that RD DOS maintains in the 17xx REU. Therefore it is highly recommended that the REU's partition be changed from native mode to foreign mode when using such programs. It was decided to auto-configure the REU as native mode since this is most useful with RD DOS. The foreign mode partition will keep RD DOS from affecting the REU.

A good understanding of RD DOS, partitioning and some experimentation will allow you to find the most suitable configuration for your application.

SECTION 1
INTRODUCTION

Getting Started

When you first receive your JiffyDOS system, you will probably be anxious to install it and to start taking advantage of the speed increases that JiffyDOS provides. Before beginning, however, please note that installing JiffyDOS requires the partial disassembly of your computer and disk drive(s). While this does not require a knowledge of electronics, it does require some manual dexterity to perform. Read through the installation instructions carefully before starting. If you do not feel confident about installing JiffyDOS yourself, refer installation to a qualified technician, or take advantage of Creative Micro Designs' installation service.

Also, please be aware that installing JiffyDOS will void any factory warranty applicable to your computer or disk drive. If you are concerned about voiding your warranty, you may want to delay installing JiffyDOS until your warranty period has expired (Commodore normally warrants their equipment for a period of 90 days).

After you have installed JiffyDOS

After you have installed JiffyDOS, please take the time to become familiar with this manual. It describes the many commands and features of your JiffyDOS system, and gives you instructions on how to use them. You will also find information that will help you get the most performance out of JiffyDOS. If you experience any problems when using JiffyDOS, please check the manual - it may provide you with the information that you need to solve the problem.

Getting Help

If you run into any problems or have any questions about the installation or operation of JiffyDOS, please feel free to contact Creative Micro Designs. We will be glad to assist in any way we can. Our address and phone number are listed below:

Creative Micro Designs, Inc.
P.O. Box 646
East Longmeadow, MA 01028
(413) 525-0023

GEORAM

Unfortunately GEORAM will not work in parallel with RAMDrive and RD DOS. However, CMD's RAMLink does have a port that enables GEORAM to be used with RL-DOS.

JiffyDOS
User's Manual

Creative Micro Designs, Inc.
PO Box 646
East Longmeadow, MA 01028
(413) 525-0023

Installation Service

Creative Micro Designs offers a JiffyDOS installation service. If you would like us to install JiffyDOS in your computer and/or disk drive, please contact us for information and prices.

What this Manual Includes

This manual gives you complete instructions on how to use your JiffyDOS system.

Section 2, "What is JiffyDOS?", describes the performance and other features of JiffyDOS.

Section 3, "Using JiffyDOS", gives you instructions on how to enable and disable JiffyDOS, how to use the JiffyDOS function keys, how to control listings and how to get the most performance out of your system.

Section 4, "The JiffyDOS Commands", provides a description of each JiffyDOS command.

The JiffyDOS Guarantee and Warranty are included at the back of this manual.

SECTION 2
WHAT IS JIFFYDOS?

JiffyDOS is an enhanced Disk Operating System (DOS) for Commodore C-64, SX-64, and C-128 computers. Programmed onto ROMs that replace the Kernal ROM(s) in your computer and the DOS ROM in your disk drive, JiffyDOS provides the speed, commands and convenience missing on stock systems. Because it is ROM-based, JiffyDOS becomes an integral part of your system, and is able to provide performance without the compatibility problems of Cartridges and other speed-enhancement systems.

Features

Uses no ports or cabling

JiffyDOS installs without any extra cabling between your computer and disk drive(s), and does not tie up the Cartridge, User or Cassette ports. This enables compatibility with all hardware devices (such as modems, memory expansions, etc.), and gives JiffyDOS an advantage over cartridge speedups and other hardware upgrades (*RapiDOS, 1541 Flash!, DigiDOS, Dolphin DOS*) which require extra cabling that plugs into one of the ports on your computer.

Built-in DOS Wedge commands

JiffyDOS includes a complete, built-in implementation of the Commodore DOS 5.1 Wedge command set. The DOS Wedge provides simple, easy-to-learn commands that eliminate the need to type complicated instructions when you need to perform common disk operations such as scratching files and formatting disks. The JiffyDOS version of the Wedge also includes a series of enhancements that make it easier to use the commands within BASIC programs.

Additional JiffyDOS commands

In addition to the standard DOS Wedge, JiffyDOS includes a number of special commands that make using your computer even easier. The JiffyDOS commands all use the familiar DOS Wedge syntax, and enable you to perform functions such as disabling the head rattle (bump) on 1541s, locking and unlocking files, listing files directly from disk, "un-NEWing" BASIC programs and dumping the screen to a printer.

Convenience features

JiffyDOS provides a full complement of convenience features that enable you to list the directory without disturbing memory; to load and run the first program on disk; and to pause, inspect and restart BASIC or JiffyDOS listings. In addition, JiffyDOS includes a full set of function key definitions that work along with the directory listing to eliminate the need to type lengthy filenames when loading, saving, or scratching files.

Does not bypass error checking

All JiffyDOS operations retain the built-in DOS error checking and correction routines that are a necessary part of any reliable data storage system. While other speedup systems and utilities (i.e. the ones boasting 10-second formats) bypass these routines, JiffyDOS provides speed without jeopardizing the integrity of your data.

Incorporates the latest Commodore upgrades

When you purchase JiffyDOS, the ROMs you receive have been programmed to include the latest Commodore upgrades. The JiffyDOS ROM for the 1571 eliminates bugs affecting Relative files, drive initialization, etc. and has also been modified to solve the problems with disk backup utilities (*Fast Hack'Em, Copy II-64/128,* etc.) caused by the Commodore upgrade. The JiffyDOS Kernal ROM for the C-64 is based on the latest version, as are the 64- and 128-mode Kernal ROMs for the C-128.

Performance

Speeds up all disk operations

JiffyDOS has been designed to speed up all operations on 1541, 1541 compatible, 1571 and 1581 disk drives. This includes the initial loading of all types of programs (including copy-protected software), saving, reading and writing files from within programs, autobooting (C-128 in 128 mode), scratching, validating, and formatting (1541 drives). JiffyDOS/128 speeds up disk operations in both 64 and 128 modes and is able to speed up the already-quick 128-mode operation of 1571 and 1581 drives.

Works within all types of software

Because of its ROM-based design, JiffyDOS is able to work from within all types of software to increase disk-access speed when you need it the most. Wordprocessors, databases, spreadsheets, programming languages, MIDI software and games all benefit from the built-in speed provided by JiffyDOS. Since JiffyDOS requires no extra RAM in your computer or disk drive and does not alter vectors, it can speed up the operation of all software that relies on the disk-access routines built into the Kernal ROM.

Speeds up access of SEQ, REL, & USR files

Another important feature of JiffyDOS is its ability to speed up the access of all types of files. Program (PRG), Sequential (SEQ), Relative (REL) and User (USR) files can all be accessed faster on JiffyDOS systems. This makes JiffyDOS effective with software that uses these file types, and provides an advantage over cartridge speedup products which improve performance only with PRG files.

Speed comparisons

The following tables illustrate the speed increases that can be obtained with JiffyDOS. Please note that the times shown are rounded off to the nearest second, and do not take into account the disk spin-up delay (approx. 1/2 second) and the time required for directory searching (which varies in relation to the size of the particular directory). Other factors may also influence the results that you obtain on your system. Refer to Section 3 for more information on these factors and for ways to obtain maximum performance when using JiffyDOS.

Compatibility

Works with virtually all software

We designed JiffyDOS to be fast.-.but not at the expense of compatibility. As a result, JiffyDOS will load and operate with virtually all software of every type (including copy-protected commercial programs). This includes programs that cannot be loaded by other speedup products such as cartridges, software-based "turbo" loaders, and other hardware-based systems. In addition, JiffyDOS is compatible with programs that utilize their own fast-access routines (such as *GEOS*) and will work with the non-standard file formats created by programs such as the *VORPAL* utility kit.

RAMDOS compatibility

JiffyDOS commands are compatible with Commodore's RAMDOS for the 1700, 1750, and 1764 RAM expansion units (REU's), as well as being compatible with the RAM operating system used by CMD in the RAMLink RAM expansion interface.

Compatible with all hardware

Because it does not use any ports, JiffyDOS can work in conjunction with all hardware devices available for C-64's and C-128's, including modems, RAM expansions, MIDI interfaces, hard disk drives, etc. JiffyDOS systems are also compatible with all serial bus devices (non-JiffyDOS disk drives, printers, printer interfaces, etc.). JiffyDOS can be installed on disk drives regardless of device number, and devices numbers can be changed on JiffyDOS drives via software or hardware with no problems. In addition, JiffyDOS-equipped disk drives can be used with stock C-64 and C-128 computers, if necessary.

Uses stock disk and file formats

All files written under JiffyDOS are identical to standard Commodore files. Disks formatted on JiffyDOS systems are identical to those formatted on stock systems. This means that you will have no problems using the disks, programs and files you create with JiffyDOS on stock systems (and vice-versa).

Can be completely switched out

In the event that a program will not load or operate properly with JiffyDOS (this should be a rare occurrence - we know of only a few heavily copy-protected games that will not load), you can switch your system back to a completely stock configuration with the flip of a switch. As an added advantage, you can use the switching system to switch JiffyDOS in or out *with power on.*

Available for all 64's, 128's and virtually all drives

JiffyDOS systems are available for the following computers and disk drives:

C-64, 64"C", SX-64, C-128, C-128D

1541, 1541"C", 1541-II, 1571, 1581
FSD-1, FSD-2, Excelerator+, Excel 2001, OC-118
MSD-1, MSD-2
Enhancer 2000
BCD/5.25, BCD/128
RF501C, RF512C, FD-168, SW71
Indus GT
CSD-1

JiffyDOS can be ordered for any combination of the above computers and disk drives. Additional drive ROMs can be ordered if you wish to speed up second, third or fourth drives. JiffyDOS allows the use of multiple dissimilar drives (i.e. 1541, 1571, 1581) whether or not they are equipped with JiffyDOS ROMs. JiffyDOS-equipped disk drives can be used interchangeably on different JiffyDOS-equipped computers (for example, a C-64 and C-128), or with stock computers. Contact Creative Micro Designs if you are interested in expanding your JiffyDOS system.

Compatibility Guarantee

Because of our confidence in the compatibility of JiffyDOS, we back our product with a 30-day, Money-Back Compatibility Guarantee. The JiffyDOS guarantee is simple: If you are dissatisfied because JiffyDOS is not compatible with any of the hardware or software that you own, simply return the unit unmodified and in good condition within 30 days of purchase for a complete refund. This offer is only valid with units purchased directly from CMD. See the back of this manual for a complete description of the JiffyDOS Guarantee.

SECTION 3
USING JIFFYDOS

ROM Switching

JiffyDOS enables your computer and disk drive(s) to be switched back to stock mode, if necessary. You will probably be able to have JiffyDOS selected all of the time. However, in rare cases, your system may have to be switched back to stock mode to allow a program to load or operate properly.

C-64, SX-64, and C-128

The normal procedure for switching Kernal selections on the C-64, SX-64, and C-128 is to first turn your computer OFF, flip the selector switch to the desired position and then power the computer back ON. After your computer is powered back on, the sign-on screen will indicate which Kernal has been selected. If the JiffyDOS Kernal is selected, the sign-on screen will read:

```
JIFFYDOS  V6.01  (C)1989  CMD
```

This may differ slightly if your version of JiffyDOS is not version 6.01. If the stock Kernal routines have been selected, the normal BASIC sign-on screen will be displayed.

1541 and compatibles

The normal procedure for switching JiffyDOS in or out on 1541 and compatible drives is to power the drive OFF, set the selector switch to the desired position and then power the drive back ON. (Note: The SX-64 does not have a separate drive selector switch.) Reading the disk drive error channel immediately after power-up will indicate which ROM selection has been made. If JiffyDOS has been selected, the message is:

```
73,JIFFYDOS  5.0  1541,00,00
```

(Note: The JiffyDOS version 6 drive ROMs have not been changed since version 5, therefore the actual version number displayed by the drive ROM is version 5). If the stock ROM routines have been selected, the message is:

```
73,CBM  DOS  V2.6  1541,00,00
```

1571 and 1581

On 1571 and 1581 drives equipped with JiffyDOS, a software switching scheme is used to select between JiffyDOS and the stock DOS. With

JiffyDOS installed, the drives sense whether the computer they are connected to is in stock or JiffyDOS mode and subsequently select the corresponding DOS routines automatically. No manual switching is required.

Switching ROMs with power on

JiffyDOS has been designed to allow ROM switching while your system is powered on. This is especially useful in cases where a heavily copy-protected program will not load with JiffyDOS selected, but will perform high-speed drive accesses properly if JiffyDOS is switched in once the program is up and running.

Precautions

The only precaution necessary for switching ROM selections on your computer or disk drive with power on is to be sure that no disk accesses are taking place at the time (switching while a disk drive is being accessed will cause the system to hang up). Take note, however, that switching Kernal ROM selections on your computer while running a program may not be 100% reliable - that is, the program running at the time may crash when the switch is toggled or may hang up when a device access is attempted (some programs will respond better than others).

Switching to the Stock Kernal while in BASIC

To switch from the JiffyDOS Kernal to the stock Kernal while your computer is in BASIC direct mode, first disable the JiffyDOS commands by using the *@Q* command and then flip the Kernal Selector Switch to the stock position.

Switching to the JiffyDOS Kernal while in BASIC

To switch from the stock Kernal to the JiffyDOS Kernal while your computer is in BASIC direct mode, first flip the Kernal Selector Switch to the JiffyDOS position and then type in one of the following commands:

```
SYS 58551          (64 mode)
SYS 65137          (128 mode)
```

Using a Tape Drive with JiffyDOS

It is not possible to use a tape drive (Commodore DATASETTE, etc.) while the JiffyDOS Kernal is selected. If tape access is attempted, an ILLEGAL DEVICE NUMBER ERROR will occur.

To use a tape drive with JiffyDOS, select the stock Commodore Kernal ROM routines first by using the Kernal Selector Switch. See "ROM Switching" earlier in this section for instructions.

Function Key Definitions

JiffyDOS offers the convenience of function keys defined with JiffyDOS Wedge commands and BASIC keywords. The function key definitions are active only in BASIC direct mode and are automatically disabled when a BASIC or ML program is run, in order to avoid conflict with any program-defined function key assignments. The JiffyDOS function key definitions are:

`f1 = @$:* <RETURN>`	Display the directory
`f3 = /`	Load a BASIC program
`f5 = ↑`	Load and run a BASIC program
`f7 = %`	Load an ML program
`f2 = @D`	List a BASIC program from disk
`f4 = @T`	List an ASCII (text) file from disk
`f6 = ←`	Save a BASIC program
`f8 = @ "S:`	Scratch a file

Using the function keys with the directory listing

The JiffyDOS function key definitions have been designed to work along with the filenames in the directory listing. Once the directory has been listed, the cursor can be moved to the line of the desired filename, the appropriate function key can be pressed, and then RETURN can be pressed when it is OK to proceed with the command.

Using the JiffyDOS function keys with the directory listing eliminates the possibility of filename misspellings. And, there is no need to enter a colon (:) after the filename or erase the filetype characters manually in order to avoid a syntax error. After the function key has been pressed, the filename can be altered if desired.

Disabling the function keys

The JiffyDOS *@F* command may be used to disable the function key definitions in 64 mode. In 128 mode this command will switch the function keys from the JiffyDOS definitions to the standard 128 definitions. Refer to Section 4, "The JiffyDOS Commands", for a complete description of the @F command.

Re-enabling the JiffyDOS function keys

The following commands are used to re-enable the standard JiffyDOS function key definitions. For more information see Section 4 of this manual.

```
SYS 58551          (64 mode)
SYS 65137          (128 mode)
```

Reprogrammable function keys (64 mode)

The function keys (C-64 or C-128 in 64 mode) can be redefined by using a BASIC program. The following program redefines the keys with the strings assigned to variables F1$ through F8$ (change these strings to the new definitions you want to assign). Please note that the keys must be assigned in the order shown in line 55 (f1, f3, f5, f7, f2, f4, f6, f8) and each definition string must end with a CHR$(0).

```
15  F1$="@$:*"+CHR$(13)+CHR$(0)
20  F2$="@D"+CHR$(0)
25  F3$="/"+CHR$(0)
30  F4$="@T"+CHR$(0)
35  F5$="↑"+CHR$(0)
40  F6$="←"+CHR$(0)
45  F7$="%"+CHR$(0)
50  F8$="@   "+CHR$(34)+"S:"+CHR$(0)
55  F$=F1$+F3$+F5$+F7$+F2$+F4$+F6$+F8$
60  BUF=820
65  FOR I=1 TO LEN(F$)
70  POKE(BUF+I-1),ASC(MID$(F$,I,1)):NEXT I
75  POKE 176,BUF-(256*(INT(BUF/256)))
80  POKE 177,INT(BUF/256):POKE 155,0
```

The variable "BUF" points to the RAM area where the new definitions will be stored (Address 820 is the Tape Buffer). This can be changed to point to any safe RAM area except for the areas under the BASIC and Kernal ROMs.

Listing Freeze

JiffyDOS features an enhanced listing freeze method that differs from V5.0. The listing freeze is especially helpful when you are scanning a long BASIC program or a listing produced by the @$ (directory), @D (list BASIC program), or @T (list ASCII file) commands. In addition, the listing freeze will work from within many commercial programs.

Scanning a listing

You can easily scan a listing by using the <CONTROL> key. Holding down <CONTROL> will stop the listing from scrolling off the screen. When you want the listing to start scrolling again, release the <CONTROL> key. While you are holding down the <CONTROL> key, you can also freeze or single-step the listing (see below).

Freezing a listing

To freeze a listing , press <CONTROL>+<S> (Press "S" while holding down the <CONTROL> key). This will stop the listing from scrolling and allow you to take your hands off the keyboard. To restart the listing, press any key except <CONTROL>, <RUN/STOP>, <SHIFT>, or the <COMMODORE> key.

Single-stepping a listing

With JiffyDOS you can "single-step" a listing. Single-stepping enables you to scroll through a listing one line at a time. To single step, first stop the listing by holding down <CONTROL>. Then, *while continually holding down the <CONTROL> key*, alternately press the "S" and "W" keys. Each time you press "W" a new line will appear on the screen. You can restart the scrolling by releasing the <CONTROL> key after you have pressed "W".

Ending a listing

JiffyDOS enables you to end (break off) any JiffyDOS or BASIC listing by pressing and holding <RUN/STOP>. You can press <RUN/STOP> while a listing is scrolling, or while it is frozen after <CONTROL>+<S> has been pressed. Note: Pressing <RUN/STOP> will work well with BASIC and JiffyDOS listings, but will not necessarily break off the listings that occur within commercial programs.

Getting Maximum Performance

As a general rule (with 1541, 1571, and compatible drives only), programs and files saved with JiffyDOS will load faster than programs and files saved on stock systems (or with JiffyDOS switched out). In some cases, re-saved files can be loaded up to 3 times faster. With programs (or files) that already load quickly with the 1571 in 128 mode, re-saving files may be the only way to obtain a discernable speed increase.

Re-saving files under JiffyDOS does not alter the file's format or contents in any way - it merely rearranges the data more efficiently on disk, which allows JiffyDOS to find each block (sector) more quickly. Re-saved files remain compatible with stock Commodore systems.

Re-writing files

To get the maximum possible speed increase when loading files from within other programs (i.e., wordprocessors, MIDI sequencers, etc.), start up the program, load in the desired file, and then re-save it. For example, to increase the load speed of a MIDI sequencer file, load your MIDI sequencing program with JiffyDOS enabled and then load the file you want to speed up into the sequencer. After the file has been loaded, simply save it back to disk. In many cases, this procedure can double the speed at which files can be loaded. Another method is to use the JiffyDOS file copier to re-write files. See the section titled "The JiffyDOS File Copier" for more information.

Changing the Sector Interleave

JiffyDOS also enables you to easily change the sector interleave on 1541 and 1571 drives. This feature makes it possible to increase the disk-access speed of particular programs that JiffyDOS seems to have little effect on. In some cases, changing the interleave may be the only way to increase disk-access performance – especially when a 1571 drive is used with a C-128 in 128 mode. For example, when a standard interleave is used, JiffyDOS does little to speed up *Paperclip III* (1571 in 128 mode). However, if the sector interleave is changed to a value of 7, file-access speed is doubled. See the section titled "Setting the Sector Interleave" for more information.

The JiffyDOS File Copier

JiffyDOS includes a built-in two-drive file copier that makes it easy to backup valuable data files, to copy files between 1541, 1571 and 1581 drives, and to copy files to and from RAM expansion units running under Commodore's RAMDOS. Although the file copier built into JiffyDOS is not the fastest one in existence, it has the following advantages over stand-alone copy utilities:

- The JiffyDOS file copier is built-in, which enables you to copy files without the hassle of loading a stand-alone utility.
- Copies PRG, SEQ, REL and most USR files.
- Saves time and trouble by automatically replacing existing files of the same name on the destination disk. (The file is first scratched, then written. Save-with-Replace is not used).
- Unlike most utilities, the JiffyDOS file copier uses the standard JiffyDOS interleave (or the interleave specified by the @*G* command), which enables you to maximize the performance of the files that you copy.

- Because the JiffyDOS copier does not rely on drive-specific routines, it can also be used to transfer non-copy-protected programs or files between any two Commodore or Commodore-compatible disk drives, including those that are not equipped with JiffyDOS. This makes it easy to move files between 1541, 1571, 1581 and compatible drives.
- The JiffyDOS file copier is compatible with Commodore's RAMDOS, which makes it ideal for transferring files to and from RAM Expansion Units.
- Files can be copied with JiffyDOS without disturbing programs currently in memory.
- The file copier commands can be used within BASIC programs, enabling you to write your own custom copy routines.
- The file copier can be used to change the file type of PRG, SEQ, and USR files.

Re-saving files

As described in the User's Manual, using a file copy utility (such as the JiffyDOS file copier) to re-save files created by the stock DOS is one of the best ways to increase performance when using 1541 and 1571 disk drives. In fact, with programs or files that already load quickly on a 1571 in 128 mode, re-saving may be the only way to obtain a discernable speed increase. The following types of files can benefit from being re-saved (copied) using the JiffyDOS file copier:

- Any file written under the stock Commodore DOS.
- Any file originally written on a 1541 drive now used on a 1571 drive.
- Any file originally written on a 1571 drive now used on a 1541 drive.

File copier commands

The following JiffyDOS commands provide you with the ability to copy files. Although only four commands are involved, they must be used properly in order to produce the desired results. Before copying any files, you should understand the function and application of each command. (For detailed information on each individual command, see the section covering JiffyDOS commands).

`@X`*`dev#`*	Sets the destination drive for copying.
`*"`*`filename`*`"`*`type`*	Copies filename to the destination drive.
`<CONTROL>+<A>`	Toggles the copy flag on/off for all files in the directory listing.
`<CONTROL>+<W>`	Toggles the copy flag on/off for a single file in the directory listing.

How to copy files with JiffyDOS

The following procedures describe the steps you will need to follow when copying files with JiffyDOS. Before using the JiffyDOS file copier for the first time, read through the procedures carefully. It is also a good idea to experiment with non-valuable disks until you are sure you can make successful copies.

Copying a single file

If you need to copy only one or two files, or do not want to disturb a BASIC program currently in memory, use the following procedure:

1. Place the source disk in the drive that you want to copy from. Place the destination disk in the drive that you want to copy to. Note: No action is necessary at this point for a source or destination that is an REU.
2. Use the `@X` command to tell JiffyDOS which drive is the destination drive. For example, enter `@X9` if the destination drive is to be device 9.
3. Use the `@#` or `<CONTROL>+<D>` commands to set the default device to the source drive. For example, enter `@#8` if the source drive is to be device 8.
4. List the directory from the source drive. You can enter `@$` or press f1. Stop the listing (press `RUN/STOP`) when you see the file(s) that you want to copy appear on the screen.
5. Cursor up to the file that you want to copy. Keep the cursor on the left side of the screen. Press the asterisk (*) key (the asterisk should appear in the leftmost column on the same line as the file you want to copy).
6. Press `<RETURN>`. The file will be copied from the source drive to the destination drive.
7. Repeat Steps 4-6 for any additional files that you want to copy.

Copying multiple files

If you need to copy more than a few files, use the following procedure:

1. Follow Steps 1-3 above under "Copying a single file".
2. Load the directory from the source drive by entering `/$`. Note: This will overwrite any BASIC program currently in memory.

   IMPORTANT: Make sure you load the directory with the `/$` command. DO NOT use the `@$` command.

   List the directory (type `LIST`). If the directory listing is longer than one screen, you can use the JiffyDOS "Listing Freeze" feature to easily locate the desired file(s). Next, use the *`<CONTROL>+<A>`* and/or *`<CONTROL>+<W>`* commands to select the files that you want to copy (see below). The asterisk (*) indicates that the file will be copied.

   Pressing `<CONTROL>+<A>` will toggle the copy flag on/off for all files in the listing, and then re-display the entire list.

   Pressing `<CONTROL>+<W>` will toggle the copy flag on/off for a single file. *The cursor must be on the same line as the file that you wish to toggle, and must be to the far left of the screen.*

   IMPORTANT: DO NOT press `<RETURN>` when the cursor is on the same line as one of the filenames. This will result in a corrupted file list, and will require that you reload the directory from disk.

3. Enter `RUN`. The file copier will start working. When all files have been copied, the cursor will return to the screen.

Using the file copier commands in BASIC programs

The following two examples illustrate how to use the file copier commands within a BASIC program.

Example 1:

```
100  @#8                          Set the source drive to device 8
110  @X10                         Set the destination to device 10
120  *"JIFFYMON"PRG               Copy PRG file "JIFFYMON"
130  *"TAX DATA"SEQ               and SEQ file "TAX DATA"
140  END                          from drive 8 to drive 10
```

Example 2:

```
100  INPUT"SOURCE DRIVE";S        Input the source drive
110  INPUT"DESTINATION";D         Input the destination drive
120  @#S : @XD                    Set the source and destination
140  INPUT"FILENAME";F$           Input the filename
150  INPUT"TYPE (P/S/R/U)";T$     Input the filetype
160  IF T$="P" THEN *F$ PRG       If file is PRG, copy it as such
170  IF T$="S" THEN *F$ SEQ       If file is SEQ, copy it as such
180  IF T$="R" THEN *F$ REL       If file is REL, copy it as such
190  IF T$="U" THEN *F$ USR       If file is USR, copy it as such
```

Changing the filetype of PRG, SEQ and USR files

The JiffyDOS file copier also enables you to change the filetype of PRG, SEQ and USR files. (Note: You cannot change a PRG, SEQ or USR file into a REL file, and you cannot change a REL file into any other type of file.) Use the following procedure to change the filetype of a PRG, SEQ or USR file:

1. Follow Steps 1-4 under "Copying a single file".

2. Cursor up to the file that you wish to change. Make sure the cursor is on the left-hand edge of the screen. Press the asterisk (*) key. DO NOT press <RETURN> yet.
3. Stay on the same line and cursor over to the filetype (PRG, SEQ, USR). Type the new filetype that you want over the one displayed on the screen, and then press <RETURN>. The file will be copied to the destination drive and will be assigned the new filetype.

Changing the filetype can also be done by using the JiffyDOS commands within a BASIC program. See below:

```
100 INPUT"SOURCE DRIVE";S          Input the source drive
110 INPUT"DESTINATION";D           Input the destination drive
120 @#S : @XD                      Set the source and destination
140 INPUT"FILE TO CHANGE";F$       Input the filename
150 INPUT"NEW TYPE (P/S/U)";T$     Input the new filetype
160 IF T$="P" THEN *F$ PRG         Make it a PRG file
170 IF T$="S" THEN *F$ SEQ         Make it a SEQ file
180 IF T$="U" THEN *F$ USR         Make it a USR file
```

Additional notes about the JiffyDOS file copier

- Whenever you copy files using the method shown above under "Copying multiple files" *make sure* that you load the directory by using the */$* command. DO NOT use the *@$* command. The *<CONTROL>+<A>* and *<CONTROL>+<W>* commands will not work properly if the directory has been loaded using *@$*.
- JiffyDOS cannot copy non-standard files such as 1581 "CBM" partition files and the special USR files created by GEOS.
- Changing a PRG file to a SEQ or USR file will not remove the first two bytes in the file (the load address). Likewise, changing a SEQ or USR file to a PRG file will not add two load address bytes to the beginning of the file.
- When JiffyDOS starts to copy a REL file, there may be a significant delay (up to about 40 sec.) while JiffyDOS interrogates the source drive to determine the record size of the particular REL file. While this operation is being carried out, the destination drive will sit inactive. Do not assume that this period of inactivity means the system has "hung up".
- After a REL file is copied, the error light will always be flashing on the source drive. This does not indicate that an error has occurred during copying.

Setting the Sector Interleave

As mentioned earlier, JiffyDOS enables you to optimize performance with some software through changing the sector interleave. The @G command is used to ease selection of the required interleave. For specific information on the @G command, see the section covering JiffyDOS commands.

Understanding sector interleave

Each time the DOS writes a file to disk, it uses a sector interleave value to lay out the physical distance (number of sectors skipped) between consecutive sectors (blocks) of a file. After a file is written, its interleave can greatly affect the speed at which the file can be read. If the space between consecutive sectors (the interleave) is too small, the DOS cannot read the next block in the file without having to wait for the disk to spin one additional revolution (one disk revolution takes 0.2 seconds). If the interleave is larger than it has to be, the DOS also spends unnecessary idle time waiting for the next sector to reach the read/write head.

Because JiffyDOS can access and transfer data faster than the stock DOS, it uses special interleave values designed to reduce access delays to a minimum. For PRG files, JiffyDOS uses an interleave that provides the fastest possible access when the Kernal Load routine is used. For SEQ, REL and USR files, JiffyDOS uses a compromise interleave value designed to perform well in most cases. On the other hand, the stock Commodore DOS uses the same interleave value for all file types. Both JiffyDOS and stock interleave values for 1541 and 1571 drives are listed in the table below:

JiffyDOS and Stock interleave values

* Depends upon track zone

The standard JiffyDOS interleave values work well in most cases. However, because of the many different ways that programs load and access data, there are times when using a different interleave value is the only way to provide maximum performance.

When to adjust the interleave

In general, if your JiffyDOS system is already performing close to its maximum capability (within a few seconds of the times shown in the performance charts in Section 2), then changing the interleave will do little to improve performance. However, if you notice that a particular program

seems to load or access data two to three times slower than is possible with JiffyDOS, then changing the interleave and resaving the program and/or data files could provide the speed you are looking for.

Files that can benefit from a new interleave

1. Any program (PRG-type) that is not loaded by the Kernal Load routine. Instead of using the Kernal Load routine, many commercial programs incorporate a loader that reads in sections of the program byte-by-byte.
2. Any PRG data file not accessed by the Kernal Load routine. Note: most programs that use PRG data files *do not* use the Kernal Load routine to access the data in the PRG file.
3. Any SEQ, REL, or USR file that is read by a slower-than-normal access routine. Unfortunately, many programmers do not take the time to optimize their file-access code (maybe because the stock Commodore DOS is so slow), or use a compiled language that is inefficient to begin with (like compiled BASIC). As a result, some commercial programs suffer from lackluster file-access performance, even after JiffyDOS is installed.

Determining the optimum interleave

To determine the optimum interleave for a particular program or data file, follow the procedure outlined below:

1. First, be sure that you have read the information above so that you understand which programs and files can benefit from an altered interleave.
2. You will need two disk drives so that the necessary files can be copied from the original disk to a new disk. The drive you will copy from is the *source drive*, the drive you will copy to is the *destination drive*.
3. Determine a starting interleave value. On 1541 drives, use 6. On 1571 drives, use 4.
4. Make a copy of the original disk. If it is a program disk, the copy should be a nibble backup, fast copy or parameter copy that is an *exact duplicate of the original with the same sector interleave*. If it is a data disk, it should contain the necessary system files, print drivers, etc. (if any), *but no actual data files*.
5. Insert the copy of the original disk into the destination drive.
6. Insert the original disk into the source drive.
7. Set the default to the destination drive. Use the `@#` or `<CONTROL>+<D>` commands. For example, enter `@#9` if the destination drive is device #9.

8. Enter the desired interleave value by using the `@G` command. For example, enter `@G5` to set the interleave to 5.
9. Set the default to the source drive. Use the `@#` or `<CONTROL>+<D>` commands. For example, enter `@#8` if the source drive is device #8.
10. Load the directory from the source drive by entering `/$`. List the directory and then use the `<CONTROL>+<A>` and/or `<CONTROL>+<W>` commands to select the program(s) or data file(s) that will be getting the new interleave (see the section covering the JiffyDOS file copier for more information on the `<CONTROL>+<A>` and `<CONTROL>+<W>` commands).

> IMPORTANT: Make sure you load the directory with the `/$` command. DO NOT use the `@$` command.

11. Tell JiffyDOS which drive is the destination drive by using the `@X` command. For example, enter `@X9` if the destination drive is device #9.
12. Copy the files from the source drive to the destination drive by entering `RUN`.
13. Test the destination disk to see if the new interleave has helped performance. If the disk is a program disk, load the program. If it is a data disk, load and run the program that will access the data, and then load the data file(s). Time the disk accesses and see if the new interleave has helped. Keep a record of all results.
14. Return to Step 5 and increase the interleave value by one. Continue this procedure until you have tried all interleave values up to and including 16.
15. Compare the timing results you recorded in Step 13 to determine the interleave value that provides the best performance.
16. Re-copy the program(s) or data file(s) using the optimum interleave value. Follow Steps 5 through 12.
17. Before continuing, make sure the standard interleave is reinstated on all drives. The safest way to do this is to completely reset your system (C64 users should power down, then up; C-128 users can press the reset button).
18. Carefully read the following procedure concerning data files that require an altered interleave.

Data files which require an altered interleave

Each time you are planning to work with data files that require an altered interleave, follow the procedure below:

1. Before loading and running the program that will access or create the data file(s), set the default to the drive that will contain the data disk.

Use the @# or <CONTROL>+<D> commands. For example, enter @#9 if the data drive is device #9.

2. Set the interleave to the optimum value for the particular program. For example, enter @G9 if the optimum value is 9. *Remember, do this for 1541 and 1571 drives only – altering the interleave on 1581 drives will have no effect on performance.*
3. If more than one drive will be accessing the data, Repeat Steps 1 and 2 for each of the additional data drives.
4. Reset the default to the drive that the program will load from. For example, enter @#8 if the program will be loaded from device #8.
5. Load and run the program that will access the data.
6. Read, write and create data files in the normal fashion from within the program.
7. After you have finished working with the program, make sure that the standard interleave is reinstated on all drives (refer to Step 17 in the previous procedure).

Important points to remember about the interleave

- Adjusting the interleave is only effective on 1541 and 1571 drives – it will not increase performance on the 1581. Adjusting the interleave will also be effective on drives that are closely compatible with the 1541 and 1571 (for example FSD, Bluechip, Amtech, Swan, Excel, Enhancer and Cardco drives). Adjusting the interleave with drives that are not as closely compatible (such as MSD and Indus drives), should be approached with caution and may in fact disturb the DOS in these drives to the point where disks and data could be corrupted.
- Files that already perform up to the maximum speed provided by JiffyDOS *will not* benefit from being re-written using a new interleave value. In fact, performance with these files will most likely suffer if a different interleave value is used. However, if a particular program loads or accesses data two to three times slower than is possible with JiffyDOS (refer to the times shown on pages 5 and 6 of the User's Manual), then changing the interleave could provide the increased speed you are looking for.
- The correct interleave for a particular program or file depends upon numerous factors and must be chosen carefully. In most cases, a trial-and-error approach must be employed to find the optimum value.
- Adjusting the interleave for a particular program or file *cannot* make up for grossly inefficient programming practices. In other words, if the software you are using is just plain slow, there just isn't much that can be done to speed up disk access. Even machine language programs can be written poorly enough to fall into this category.
- Adjusting the interleave may not do much for slow-running languages like BASIC. A good example is a loop that reads a file one byte at a time using the GET# statement. Because the BASIC interpreter takes so long to execute each instruction, far too much time passes between the actual reading of each byte. Complied BASIC is better, but not as much as you would expect. While adjusting the interleave may help speed up file access from within BASIC, it will most likely not provide the maximum speed that JiffyDOS is capable of.
- Some programs utilize custom file access routines that completely bypass the Kernal ROM. GEOS and Flight Simulator II are two examples. Neither JiffyDOS nor an altered interleave can help the performance of such programs.
- Files saved under JiffyDOS using a standard or altered interleave will most likely load *slower* on stock systems.
- Be careful if you have both 1541 and 1571 drives. The optimum interleave for a 1571 drive may cause severely decreased performance on a 1541 drive (1571's can read data much faster than 1541's). If you have disks that will be used on *both* 1541 and 1571 drives, then use the optimum interleave for the slower 1541 drive.

If you're not getting top Performance

When using certain programs, you may feel that JiffyDOS is not increasing performance as much as it should be. In these cases, the problem is most likely related to the particular software in use. There are also some considerations to note when using 1571 drives.

Programs with built-in fast loaders

Some programs utilize their own built-in fast loaders. Some examples are *GEOS*, the *Mastertracks* MIDI sequencer, *Fast Hack'Em*, and disks created by certain types of "freezer" cartridges. These programs contain their own DOS which bypasses the disk-access routines in the Kernal and DOS ROMs. Since JiffyDOS resides in these ROMs, any program which bypasses the disk-access routines renders JiffyDOS ineffective. As a result, JiffyDOS can provide little, if any, performance gains with such programs.

In these cases, better performance may be achieved only by alterering the software so that it uses the disk-access routines provided in the Kernal ROM, or to alter the interleave as outlined previously. Altering the software itself can be a difficult job, even if you have the programming know-how. Another option is to contact the software manufacturer to see if any versions

of the program exist that do not incorporate the built-in fastloader, or if there is an easy way to disable it. If you own the *Mastertracks* MIDI sequencer, contact Creative Micro Designs to obtain a free conversion utility which disables the built-in fastloader and enables JiffyDOS to take charge of disk accesses.

BASIC programs

Programs written in BASIC will produce mixed results with JiffyDOS because of the slowness of the BASIC interpreter. The interpreter spends most of its time figuring out what each instruction is and what it should do with it - making disk access time insignificant when compared to this software overhead. As a result, the speed increases you notice when using JiffyDOS will vary according to the way the BASIC program was written.

JiffyDOS performs best when loading another program from within BASIC (i.e. `LOAD"TEST.ML",8,1`). Because the entire file is accessed by a single instruction (the `LOAD` command), JiffyDOS can perform at 100% efficiency. On the other hand, reading a file byte-by-byte by means of the `GET#` instruction is the least efficient method of accessing a file. Because of the amount of software overhead involved in interpreting the `GET#` instruction and the other instructions in a typical `GET` loop (and because of the disk-access characteristics of `GET#`), you will probably notice no better than a 2x speed increase with JiffyDOS. Whenever possible, use the `INPUT#` instruction to read files. `INPUT#` will perform much better than `GET#` - especially if you read in a long string each time `INPUT#` is executed (the longer the string, the better the performance).

Compiled BASIC (i.e. programs compiled using *Basic-64* or *Blitz!*) will perform better with JiffyDOS than uncompiled BASIC (the elimination of interpreter overhead is the main reason). However, the degree of speed increase will still depend upon how the original program was written. See the above paragraph for a discussion of the `GET#` and `INPUT#` instructions.

Machine-language programs

JiffyDOS performs best when disk accesses are handled by efficient, well-written machine-language routines (JiffyDOS performance specifications are based on results obtained using ML routines). However, not all machine-language routines are efficiently written - which means that the speed increases you experience using JiffyDOS can vary greatly from program-to-program.

Fragmented files

"Fragmented" files have their data scattered across a disk in a non-orderly fashion. This usually occurs on disks that have had a number of files

"scratched", or erased. When a file is saved to such a disk, the DOS first fills up the empty areas left by the scratched files. If the file being saved is long, it may be written to many of these randomly-located areas. When it comes time to read the file, the DOS is forced to search across a wide area of the disk in order to retrieve all the data, which can add significantly to the amount of time it takes to read the file.

To eliminate file fragmentation, copy the files from the fragmented disk to an empty disk. You can transfer the files using the JiffyDOS file copier or a file-copy program. See "Getting Maximum Performance" earlier in this section for more information.

Sprites

"Sprites" are graphic display items that can be moved around quickly and easily on screen. Although sprites are used primarily in games (Pac-Man is a sprite), they are also found in other types of programs (the *GEOS* pointer is also a sprite). While it seems unlikely that a screen graphic could affect disk-access performance, sprites present a real problem for serial-bus communications routines. Sprites are hardware-generated by the VIC-II chip, and displaying them requires the VIC-II to "steal" cycles from the 6510/8510 microprocessor. Cycle stealing makes it impossible to use high-speed software timing loops to transfer data over the serial bus (JiffyDOS uses timing loops). Even the stock Commodore C-64 DOS is prone to hang up when sprites are displayed during serial I/O.

The solution to the problem is to disable sprites (shut them off) whenever serial bus communications are taking place. This is the approach used by *GEOS* (you'll notice the pointer disappear during disk accesses). Unfortunately, few programs imitate *GEOS* in this regard. Because of such programs, JiffyDOS is forced to disable sprites on its own (Commodore also does this in their 128-mode DOS). When JiffyDOS has to disable sprites for each byte read or written over the serial bus, data transfer slows down to the turtle-like speed of stock systems (this slowdown also occurs, by the way, with Commodore's 128-mode DOS).

Unfortunately, there is not much you can do if sprites are causing JiffyDOS to slow down (unless, of course, you have written the offending program yourself and can change it).

1571 drives

If you are using a JiffyDOS-equipped 1571 drive with a C-64, SX-64, or a C-128 in 64 mode, you will obtain maximum performance *only if the 1571 is in 1541 mode*. With JiffyDOS installed, your 1571 can access (and format) both sides of a disk in 1541 mode - making it unnecessary to switch to 1571 mode to benefit from the increased dual-side storage. This means:

1. You should not use the "U0>M1" command to switch into 1571 mode when using a C-64, SX-64 or C-128 in 64 mode (this will reduce performance). C-64/SX-64 owners: Note that the 1571 is automatically in 1541 mode upon power-up.

2. When switching from 128 mode to 64 mode on a C-128, *use the reset switch and hold down the Commodore key*. Do not use the "GO64" command. Using GO64 will leave the 1571 in 1571 mode, resulting in reduced performance in 64 mode.

Note: If you must use the GO64 command, sending the `"U0>M0"` command to the 1571 disk drive will place it in 1541 mode, ensuring best performance.

If a Program won't Load or Operate

Although we have made every effort to ensure compatibility with as much software as possible, you may occasionally run across a program that will not load or operate properly with JiffyDOS. Loading problems are almost always caused by extravagant copy-protection schemes that rely on stock head step rates, serial bus timing, and particular ROM versions (these copy protection schemes are the same ones that cause loading problems on 1571 and 1541C drives). Unfortunately, it would be necessary to reduce the performance of JiffyDOS in order to provide compatibility with these programs.

Another very rare type of program will load OK under JiffyDOS but will not operate properly once loaded. This type of problem is caused almost exclusively by bad programming practices (i.e. the indiscriminate use of ROM routines without regard to normal entry points, jump tables, etc.). This misuse of ROM routines has become so widespread that Commodore is now very reluctant to change (even fix or improve) their own ROMs for fear of introducing compatibility problems. Since JiffyDOS relies on carefully selected ROM changes to enhance performance, undoing any of these alterations to provide compatibility with just one program would unfortunately reduce performance, features, or both.

What to do if a program won't load

JiffyDOS includes a switch which will return your system to a 100% stock configuration to ensure compatibility with the few programs that do present problems. Refer to "ROM Switching" earlier in this section for information on how to switch JiffyDOS out in order to provide compatibility with problem programs.

Using JiffyDOS with RAM Units

The JiffyDOS commands support Commodore's RAMDOS for the 1700, 1750 and 1764 RAM Expansion Units in both 64 and 128 modes. Please read the following information carefully before using an REU with JiffyDOS.

Recommended RAMDOS Versions

The following versions of Commodore's RAMDOS are required for proper operation with JiffyDOS:

64 mode:	RAMDOS V4.2 (or higher)
128 mode:	RAMDOS V4.3 (or higher)

Using earlier RAMDOS versions will subject you to the bugs that they contain and to incompatibilities with certain JiffyDOS commands. If you do not have the correct version of RAMDOS, it can be obtained from Q-Link, a User's Group, bulletin board or friend. Note: All versions of Commodore's RAMDOS are in the public domain.

JiffyDOS commands that conflict with RAMDOS

The following JiffyDOS commands are incompatible with Commodore's RAMDOS and should not be used:

`@L:filename`	(Lock a file)
`£filename`	(Load and run a machine-language program)

Starting up RAMDOS on your JiffyDOS system

To use RAMDOS along with JiffyDOS, do the following:

1. Power up or reset your system with JiffyDOS active (the power-on screen message will indicate if JiffyDOS is switched in).
2. Start up RAMDOS as you would normally. DO NOT load the DOS Wedge supplied on your RAMDOS diskette – it will disable the JiffyDOS commands.
3. Use the JiffyDOS commands as you would normally to access the REU. Remember to avoid the JiffyDOS commands that are not compatible with RAMDOS (see above).

The limitations of RAMDOS

Commodore's RAMDOS relies on software vectors to operate. Since many commercial programs reset these vectors, they effectively disable RAMDOS and render the REU useless. Because of the design of RAMDOS and the REU, JiffyDOS cannot overcome this limitation and provide increased

commercial compatibility with RAMDOS. So, although JiffyDOS will make using your REU easier, it cannot by itself extend its usefulness with commercial software. Future CMD products are planned which will overcome these problems and make your REU useable with nearly all software.

Using JiffyDOS Commands with BASIC

Whenever JiffyDOS executes a command, it opens up a 255-byte buffer at the top of BASIC. If you have a BASIC program in memory that takes up all (or almost all) of the available BASIC memory, an OUT OF MEMORY error may occur when the buffer is opened. If this problem occurs with a particular program you are working on or using, you can do two things to overcome the memory shortage: 1) If you are in direct mode, enter CLR. In most cases, this will free up enough memory for the buffer. 2) Shorten the offending program, or cut down on its use of arrays and variables.

SECTION 4
THE JIFFYDOS COMMANDS

JiffyDOS offers the convenience of a complete, built-in implementation of the Commodore DOS 5.1 Wedge command set. Also included are a set of additional commands that provide functions that are not accessible on stock systems or through the standard DOS Wedge. All JiffyDOS commands can be entered in BASIC direct mode and/or used from within BASIC programs.

Command Descriptions

This section gives a complete description of each JiffyDOS command. The commands are listed by their function, along with the syntax of each form of the command. Examples are provided after the command descriptions to illustrate the use of each command in its basic and optional forms. If no example is given, the only form of the command is as listed in the syntax line. Here is a brief description:

`...`	Indicates a command in which the last parameter given may be repeated. NOTE: The maximum length of a command string is 41 characters.
`[ ]`	Used to enclose information which is optional to the command syntax. The brackets themselves are not part of the command.
`filename`	Indicates where a filename should be placed in a command. Other names are used to indicate filenames as well such as *`oldfile`*, *`file1`*, and *`backup`*.
`dev#`	Used to indicate where a device number should be placed in a command. Legal device numbers are usually 4 or 5 for printers, and 8 through 11 for disk drives.
Direct:	Shows the syntax of the command when used in direct mode.
Program:	Shows the syntax of the command when used in a program. This syntax may usually be used in direct mode as well, and in many cases, must be used in order to take advantage of the default device override feature.

Parameters printed in *`italics`* are not literal and should be replaced with the proper information when using the command.

Parameters printed in plain text are literal and must be included in the command.

A *drive number* may optionally be used with most drive commands. This is useful when using dual drives such as the MSD-2 or Commodore 4040. To indicate a drive number (0 or 1), place the drive number just prior to the colon (:) in the command string. If using a command with quotes, place the device number and colon within the quotes.

Setting the Default Device

Two commands are available for switching the JiffyDOS default device on multiple-drive systems: The standard DOS Wedge command @#, and the convenient JiffyDOS <CONTROL>+<D> device toggle.

@#

Direct: @#*dev#*

Program: same

The @# command is used in multiple-drive systems to specify the default disk drive device number for all JiffyDOS commands. When this command is entered, an attempt is made to access the drive with the device number specified within the command. If the access succeeds, the specified drive becomes the new default drive. If the access fails, a DEVICE NOT PRESENT error will be displayed, and the default drive will remain unchanged.

@#9	Sets the default drive device number to 9 (if device 9 is present).

<CONTROL>+<D>

The <CONTROL>+<D> command is used on multiple-drive systems to switch the default disk drive device number for all JiffyDOS commands. Pressing <CONTROL>+<D> (holding down the <CONTROL> key and pressing D at the same time) will switch the default device assignment and display the new device number on the screen. For example, if device 8 is the current default, <CONTROL>+<D> will make device 9 (if present) the default, and display a "9" on the screen. If device 9 is the current default, pressing <CONTROL>+<D> will make device 8 the default, and display an "8" on the screen. <CONTROL>+<D> offers a much more convenient means of changing default devices than the @# command. The <CONTROL>+<D> command is only available in direct mode.

Displaying the Directory

Direct: @$[:*filename*]

Program: @"$[:*filename*]"[,*dev#*]

JiffyDOS provides a quick, convenient way to display disk directories without overwriting programs stored in your computer's BASIC memory. JiffyDOS directories can be easily printed, suspended or ended at any time.

The @$ command displays the disk directory. Typing @$ with no filename displays the entire directory. @$ followed by a colon and a filename will display the specified file in the directory listing (if it exists). A selective listing can be displayed by using pattern matching or wild cards as part of the filename. The directory can be printed by entering the @P command prior to issuing the @$ command, and after the listing has been completed, output can be restored to the screen by issuing another @P command.

Note: A directory listing produced by the @$ command can be paused or suspended at any time by using the commands described in the section titled "Listing Freeze".

@$	Displays the entire directory.
@$:TEST	The directory listing will display the file TEST if it exists on disk.
@$:T*	All files having a "T" as the first letter in their filename will be displayed in the directory listing.
@P @$ @P	Lists the directory to a printer. Output is then restored to the screen.

Reading the Disk Drive Error Channel

Direct: @["",*dev#*]

Program: same

JiffyDOS makes it easy to find out what has happened when the error light on your disk drive starts flashing. Instead of having to type in a program (as you must on stock systems), you can enter a single character to display the error messages output by your disk drive.

The @ command is used to read and display the disk drive error channel. This is useful in determining what type of error has occurred when the red error light on the disk drive is flashing (you will also be pleased to find that

issuing the @ command will also shut off the distracting flash of the error light). If @ is typed when the error light is not flashing, the drive "OK" message will be displayed (`00, OK,00,00`).

If the @ command is entered when the system (or your disk drive) has just been powered up or reset, the disk drive's DOS type and version number will be displayed.

Loading BASIC Programs

JiffyDOS provides two commands for loading BASIC programs.

Load a BASIC program

Direct: `/filename`

Program: `/"filename"[,dev#]`

The / command provides a shorthand method of loading a BASIC program. Typing `/filename` is identical to entering the BASIC command `LOAD"filename",n` (where n is the default drive device number).

Load and RUN a BASIC program

Direct: `↑filename`

Program: `↑"filename"[,dev#]`

The ↑ command loads and runs a BASIC program. Typing `↑filename` is the same as entering the BASIC commands `LOAD"filename",n` (where n is the default drive device number) and `RUN`.

Saving BASIC Programs

Direct: `←filename`

Program: `←"filename"[,dev#]`

The JiffyDOS ← command saves a BASIC program. Typing `←filename` is equivalent to entering the BASIC command `SAVE"filename",n` (where n is the default drive device number).

Loading Machine-Language Programs

JiffyDOS provides two powerful commands for loading machine-language programs.

Load a machine language program

Direct: `%filename`

Program: `%"filename"[,dev#]`

The % command loads a machine language file without resetting any BASIC pointers. Typing `%filename` is the equivalent of entering the command `LOAD"filename",n,1` (where n is the default drive device number), except that the BASIC program pointers are not reset as a result of the load. This avoids OUT OF MEMORY errors, and also means that the BASIC program pointer is not reset to the beginning of the current program. Thus, this command provides a way to load ML programs from within a BASIC program without resorting to the usual contortions. For example, use a routine similar to the following to load a number of machine language modules:

```
10 %"ML1":%"ML2":%"ML3"
```

Load and execute machine language program

Direct: `£filename`

Program: `£"filename"[,dev#]`

The £ command loads and then runs a machine-language program. Typing `£filename` is a much more convenient alternative to entering `LOAD"filename",n,1` and then doing a `SYS` to start the program. At the beginning of a load, the £ command finds the starting address of the ML file and then begins execution at that address when the program has finished loading. Note: While this command will work properly with most machine language programs, it will not work with programs that have entry points that differ from their load addresses.

Loading and Running the First Program on Disk

`<Shift>+<RUN/STOP>`

The *<Shift>+<RUN/STOP>* key combination can be used to load and run the first program on disk. Using this command is identical to entering the BASIC commands `LOAD":*",8,1` and `RUN`. This provides a quick, convenient way of starting up most commercial software programs. In 64 mode the `<Commodore>` key may be used instead of `<Shift>`.

Verifying Programs

Direct: `'filename`

Program: `'"filename"[,dev#]`

The ' (apostrophe) command verifies a file in memory against a file on disk. Using this command provides a shorthand method of entering the BASIC command `VERIFY"filename",n,1` (where n is the default drive device number). This command will work with all machine language files and with all BASIC programs that have been saved under the current BASIC environment (i.e. a BASIC program saved on the C-64 will verify properly on a C-64 but not on a C-128 in 128 Mode).

Listing BASIC Programs from Disk

Direct: `@D:filename`

Program: `@D"filename"[,dev#]`

The *@D* (DLIST) command enables a BASIC program to be listed to the screen or to a printer directly from the disk without loading it into memory first. The file listing can be directed to a printer by issuing the `@P` command prior to using this command. After the listing has been completed, output can be restored to the screen by entering the `@P` command once again. For legible results, take care to use `@D` to list only BASIC programs.

Note: A listing produced by the `@D` command can be paused or suspended at any time by using the commands described in the section titled "Listing Freeze".

`@D:GAME`	A listing of the program GAME will be sent from disk to the screen.

`@P`
`@D:LOTTERY`
`@P`
	A listing of the program LOTTERY will be output from disk to a printer (device #4). Output is then restored to the screen.

Listing ASCII Files from Disk

Direct: `@T:filename`

Program: `@T"filename"[,dev#]`

The most common use of the *@T* (TYPE) command is to list an ASCII text file (such as a wordprocessor file) to the screen or to a printer. When using `@T`, care should be taken to list ASCII files only - otherwise, the results may be unpredictable (for example, a BASIC program will display

garbage if listed to the screen or to a printer via `@T`). The default output device for the `@T` command is the screen. To list a file to the screen, enter `@T:filename`. To print a listing, issue the `@P` command prior to using the `@T` command. When the printout is complete, you can restore output to the screen by entering `@P` once more.

Note: A listing produced by the `@T` command can be paused or suspended at any time by using the commands described in the section titled "Listing Freeze".

`@T:LETTER`	Lists the file LETTER to the screen.

`@P`
`@T:APPOINTMENTS`
`@P`
	Lists the file APPOINTMENTS to a printer. Output is restored to the screen at the completion of the command.

"Un-NEWing" a BASIC Program

Direct: `@O`

The JiffyDOS *@O* (Old) command can be used to recover a BASIC program that has been accidentally "NEWed." Entering `@O` will also recover a BASIC program lost when the computer is reset. Note that the Un-NEW command is only effective if no new BASIC lines have been entered since the original program was lost.

Initializing the Disk Drive

Direct: `@I`

Program: `@"I"[,dev#]`

The *@I* (INITIALIZE) command initializes the disk drive. Initialization causes the drive to read the BAM (block availability map) and ID from a diskette and store this information in its internal memory. Initialization also clears the error channel and turns off the error LED on the drive (if it is flashing). At times (i.e. after a DRIVE NOT READY error), it will be necessary to initialize the disk drive before it can be used to perform any further operations. Remember that the `@I` command clears the error status of the drive, so if you wish to read the drive error channel after an error has occurred (see the `@` command), do so before using this command. Note that it is a good idea to initialize the disk drive every time you insert a diskette into the drive.

Resetting the Disk Drive

Direct: `@UJ`

Program: `@"UJ"[,dev#]`

The *@UJ* command is used to reset the disk drive. Note that `@UJ` may not work properly with older 1541 drives. For this command to work with older 1541's, the alternate syntax `@U:` may be required.

Validating Disks

Direct: `@V`

Program: `@"V"[,dev#]`

The *@V* (VALIDATE) command is used to free unused blocks on a diskette. After a diskette has been in use for a while, more blocks may be free than the BAM and directory indicate. Primarily, this happens when unclosed files have been scratched. Using the `@V` command will free space on a diskette available for use. Note that this command may take some time to complete, depending on the amount and size of the files located on the particular diskette.

Formatting Disks

The *@N* (NEW) command is used to format a diskette. There are three versions of this command to cover long, short, and extended formatting of the diskette.

Note: Using inferior media can cause errors during formatting. JiffyDOS is less forgiving in this respect than the stock DOS, but when a formatting error occurs it should indicate to you that the media may not be reliable.

"Long" NEW

Direct: `@N:diskname,ID`

Program: `@"N:diskname,ID"[,dev#]`

This version of the `@N` command is necessary when formatting a diskette for the very first time. *Diskname* is any name you wish to assign to the disk and can be up to 16 characters long (the diskname appears at the top of directory listings). The *ID* can be any 2 characters and is written to the directory along with the diskname and also to all header blocks on the diskette. Remember to include the comma between *diskname* and *ID* when using this form of the command.

"Short" NEW

Direct: `@N:diskname`

Program: `@"N:diskname"[,dev#]`

This version of the `@N` command performs a "short" NEW on a diskette that has already been formatted. A short NEW clears the directory and BAM of an already-formatted diskette and is much quicker than completely reformatting.

"Extended" NEW for 1571 from 64 mode

(JiffyDOS-equipped 1571 drives only)

Direct: `@N2:diskname,ID`

Program: `@"N2:diskname,ID"[,dev#]`

This version of the `@N` command is used exclusively to format both sides of 1571 disks when the 1571 is in 1541 mode (the stock Commodore DOS allows only one side of a diskette to be formatted in 1541 mode). This command helps take advantage of the double-side 1541-mode operation provided by JiffyDOS. Note that the 1571 is in 1541 mode upon power-up, and will only switch to 1571 mode if used with a C-128 in 128 mode, or if the `U0>M1` command is issued. The `@N2` command is effective only on 1571's equipped with JiffyDOS - using this command on other drives will result in a syntax error (no formatting will take place).

Examples:

`@N:NEWDISK,00`	Completely formats (NEWs) a diskette and assigns the name "NEWDISK" and the ID code "00" to the diskette.
`@N:BACKUP`	Performs a short NEW (clears the directory and BAM) and assigns the name "BACKUP" to the diskette.
`@N2:TWOSIDE,01`	Formats both sides of a 1571 disk when the 1571 is in 1541 mode. Assigns the name "TWOSIDE" and the ID code "01" to the diskette.
`@"N:NEWDISK,00",9`	Completely formats a diskette in the drive assigned as device number 9 with the name "NEWDISK" and the ID code "00".

Disabling the 1541 "Head Rattle"

Direct: @B (JiffyDOS/64 only)

Program: same (JiffyDOS/64 only)

The *@B* (BUMP) command can be used to disable the 1541 head rattle (bump) routine which occurs when certain disk errors are encountered. Entering @B will disable the head rattle. If you wish to re-enable the head rattle, you may do so by issuing the *@UJ* command. This command does not support the default device override function.

Note: To ensure accurate formatting, @B will not disable the head bump at the start of a disk NEW (format).

Copying files

The JiffyDOS file copier consists of four specialized commands that enable you to make quick, convenient copies of non-copy-protected programs or data files. See the section on using the JiffyDOS file copier for step-by-step procedures on how to apply these commands.

Set Destination Device

Direct: @X*dev#*

Program: same

The *@X* command sets the destination drive for copying files. The device number specified in the command can be that of any disk drive on the serial bus, or the device number of a RAM expansion unit.

@X10	Set the destination drive to device 10.

Copy file from source to destination

Direct: *"*filename*"*filetype*

Program: same

The * (asterisk) command copies a file to the destination drive specified in the *@X* command (see above). The filetype specified by this command can be PRG, SEQ, REL or USR.

*"TEST"PRG	Copies the PRG file "TEST" to the destination drive specified in the last *@X* command that was issued.

<CONTROL>+<A>

The *<CONTROL>+<A>* command toggles the copy flag for all files in the directory listing. For this command to work properly, the directory must first be loaded into memory using the */$* command. When you press <CONTROL>+<A>, the copy flag for each file in the directory will be toggled, and then the directory will be re-listed to the screen with the new copy status for each file displayed. A file marked for copying will have an asterisk (*) next to it in the directory listing. After you have marked the files that you want to copy, the actual process of copying the files is started by entering the BASIC RUN command .

<CONTROL>+<W>

The *<CONTROL>+<W>* command toggles the copy flag of a single file in the directory listing. For this command to work properly, the directory must first be loaded into memory using the */$* command. To select the file that you wish to toggle, move the cursor to that file in the directory listing and press <CONTROL>+<W>. *The cursor must be kept to the leftmost edge of the screen.* A file marked for copying will have an asterisk (*) next to it in the directory listing. If the directory listing is longer than one screen, you can use the JiffyDOS "Listing Freeze" feature to easily locate the desired file(s). You can redisplay the entire directory listing at any time by entering the BASIC LIST command. After you have marked the files that you want to copy, the actual process of copying the files is started by entering the BASIC RUN command .

Changing the Sector Interleave

JiffyDOS provides a means for adjusting the interleave (intersector gap) for files it writes. Adjusting the sector interleave can provide faster program loading and file-access performance in instances where the standard JiffyDOS interleave values are not appropriate. Adjusting the interleave can be one of the best ways to improve performance when a 1571 disk drive is used on a C-128 in 128 mode. See the section on adjusting the sector interleave under "Getting Maximum Performance" for step-by-step procedures on how to apply this command.

Gapsize

Direct: @G*gapsize*

Program: same

The *@G* command changes the sector interleave (intersector gap) of the current JiffyDOS default disk drive to the value specified by gapsize. Valid gapsize values range from 0 to 16. Since no range checking is done by this

command, be sure that you do not exceed the above limits (0-16), or else unpredictable drive operation may result during writes to disk. If you find that you have entered an illegal value, simply re-enter the command with the correct value. To reinstate the default JiffyDOS interleave, enter a gapsize of 0. Remember, changing the sector interleave is only effective on 1541, 1571 and compatible drives – do not try to change the interleave of 1581 or MSD drives. Also, please note that the default device override feature cannot be used with this command.

`@G8`	Set the sector interleave of the default device to 8.
`@G0`	Re-establish the standard JiffyDOS interleave.

Combining Files and Creating Backups

Direct: `@C:newfile=oldfile[,file2]...`

Program: `@"C:newfile=oldfile[,file2]..."[,dev#]`

The *@C* (COPY) command enables you to duplicate a file. `@C` can also be used to combine two or more text or data files into a single file. Programs may not be combined in this manner. Note that `@C` will duplicate a file under a different name on the same diskette, but will not copy or combine files from one diskette to another, or from one drive to another.

`@C:BACKUP=ORIGINAL`	Creates a new file named BACKUP which is an exact duplicate of the file named ORIGINAL.
`@C:DATA=BITS,BYTES`	Creates a file named DATA which is the combination of the files BITS and BYTES.

Renaming Files

Direct: `@R:newname=oldname`

Program: `@"R:newname=oldname"[,dev#]`

The *@R* (RENAME) command can be used to rename a file on disk. When using the `@R` command, remember that the new name for the file is entered first (to the left of the = sign) and that the existing name of the file is entered second (to the right of the = sign).

`@R:REPORT=NOTES`	Renames the file NOTES as REPORT.

Scratching (Deleting) Files

Direct: `@S:file1[,file2]...`

Program: `@"S:file1[,file2]..."[,dev#]`

The *@S* (SCRATCH) command can be used to delete (scratch) a file or a number of files from a disk. Files can be scratched one at a time (by entering the exact filename) or in groups, by specifying filenames explicitly or by using pattern matching and/or wild cards. The error channel may be is read at the completion of the `@S` command by issuing the `@` command, but will not otherwise be displayed. If used after scratching, the number of files scratched will be displayed. For example, the following message indicates that two files have been scratched:

```
01, FILES SCRATCHED,02,00
```

This message indicates that no files have been scratched:

```
01, FILES SCRATCHED,00,00
```

Examples of using the `@S` command:

`@S:TEST`	Scratches the file named TEST.
`@S:*`	Scratches all files from a disk.
`@S:T*`	Scratches all files that begin with the letter T.
`@S:?T`	Scratches all files with names two characters long that have a "T" as the second character.
`@S:BITS,BYTES,WORDS`	Scratches the files BITS,BYTES and WORDS

Locking and Unlocking Files

Direct: `@L:filename`

Program: `@"L:filename"[,dev#]`

The *@L* (LOCK) command locks and unlocks files. Once a file has been locked, it cannot be accidently scratched and is marked with a "<" at the end of its directory entry (i.e. `32 "TESTFILE" PRG<`). The `@L` command works as a toggle - that is, if `@L` is used on an unlocked file, the file will be locked. If `@L` is used on an already locked file, the file will be unlocked.

Note: The LOCK command (@L) will not work with 1581 or MSD disk drives which are not equipped with JiffyDOS.

`@L:TESTPROGRAM`	Locks the file TESTPROGRAM (if TESTPROGRAM is not currently locked).
`@L:TESTPROGRAM`	Unlocks the file TESTPROGRAM (if TESTPROGRAM is currently locked).

Directing Output to a Printer

Direct: `@P[,dev#][,secondary address]`

Program: same

The *@P* (Printer output toggle) command makes it easy to send BASIC program listings, directory listings, or output from the JiffyDOS @D and @T commands to a printer. Entering @P before issuing a command that normally produces a listing to the screen will result in the screen output being redirected to your printer (the default is device #4, secondary address=0). If you wish to specify a different printer device number and/or secondary address, include the device number and secondary address in the @P command as shown in the examples below. To restore output to the screen after using the @P command, simply enter @P again. Normally, the secondary address of 0 is used for uppercase/graphics printing, and a secondary address of 7 is used for uppercase/lowercase printing.

`@P`	Send screen output to printer (device #4, secondary address 0)
`@P,5`	Send screen output to printer (device #5, secondary address 0)
`@P,5,7`	Send screen output to printer (device #5, secondary address 7)
`@P`	Restore output to screen (assuming @P has already been used to redirect screen output to a printer).

Printing the Screen

<CONTROL>+<P>

The *<CONTROL>+<P>* (PRINT SCREEN) command outputs the current text screen to a printer. Pressing <CONTROL>+<P> (holding down the <CONTROL> key and then pressing "P") will cause the text displayed on the screen to be sent to a printer. This function automatically detects the current screen mode in order to print from the proper character set. The printer must be configured as device 4.

Disabling the JiffyDOS Function Keys

Direct: @F

Program: same

The *@F* (FUNCTION KEY) command is used to disable the JiffyDOS function key definitions. Typing @F with the function keys enabled (as they are upon power-up) will disable them. In 128 mode this will cause the standard 128 function key definition to be activated. The JiffyDOS function key definitions are active only in BASIC direct mode and are disabled when programs are run in order to avoid conflict with any program-defined function keys assignments.

Re-Enabling the JiffyDOS Function Keys

`SYS 58551`	(JiffyDOS/64)
`SYS 65137`	(JiffyDOS/128)

The *SYS* commands are used to re-enable the JiffyDOS function key assignmentss. This can be necessary if the commands have been previously disabled via the @F command, or after a program has been run.

`SYS 58551`	Will restore the JiffyDOS/64 function keys and commands on a C-64, SX-64 or C-128 in 64 mode.
`SYS 65137`	Will restore the JiffyDOS/128 function keys on a C-128 in 128 mode.

Disabling the JiffyDOS Commands

Direct: @Q

Program: same

The *@Q* (QUIT) command disables all JiffyDOS commands and function key definitions and reinstates all stock BASIC vectors. The disk drive fast-access routines are not disabled, however, and will continue to operate at full speed.

Re-Enabling the JiffyDOS Commands

`SYS 58451`	(JiffyDOS/64)
`SYS 65137`	(JiffyDOS/128)

These *SYS* commands are used to re-enable the JiffyDOS commands. This can be necessary if the commands have been previously disabled via the `@Q` command, if the JiffyDOS Kernal has just been switched in while computer power is ON, or if an applications program has altered the BASIC tokenization or execution vectors. In any of these cases, the JiffyDOS commands will not be available.

`SYS 58451`	Restores the use of all JiffyDOS/64 commands on a C-64, SX-64 or C-128 in 64 mode. This will not restore the JiffyDOS function key definitions.
`SYS 65137`	Restores the use of all JiffyDOS/128 commands on a C-128 in 128 mode.

Special Command Features

Using the Commands in Direct and Program Modes

All JiffyDOS commands may be used in BASIC direct mode and/or from within BASIC programs. When using the commands within programs, many of the command/filename strings must be enclosed in quotes. For example, `@S:FILENAME` is acceptable in direct mode only, while `@"S:FILENAME"` is the required form for program mode. The following are examples of using commands in program mode. Be sure to check the command syntax given in the previous reference section if you are uncertain how to use a command in program mode.

Examples of program-mode command syntax:

`10 @"$:*"`	Display the directory
`20 @"S0:filename"`	Scratch a file
`30 @#9`	Set the default device to 9
`40 @"D:filename"`	List a BASIC program from disk
`50 @"I:"`	Initialize the default drive

Drive Specification

All commands which perform a disk operation may specify a drive number (0 or 1) within the syntax of the command. This is useful if a dual-drive unit (i.e. MSD-2 or Commodore 4040) is sharing the serial bus with a JiffyDOS system. To specify the drive number, insert the 0 or 1 in front of the colon in each command.

Examples:

`@$1:*`	List the directory from drive 1
`@S0:filename`	Scratch a file from drive 0

When addressing 1541 and compatible drives, the 0 or 1 is not required, but a 0 may be used if desired.

Enhancements to the DOS Wedge

JiffyDOS offers these three major enhancements to the standard DOS Wedge command format.

Command Chaining

In program mode, the JiffyDOS commands may be chained with BASIC commands or other JiffyDOS commands within the same program line.

Examples:

`10 @#9:@"S:TEST":@#8`	Switch default to device 9, scratch a file, and then switch the default back to device 8.
`20 @"$:*":PRINT:@`	Display the directory, print a blank line, and display the error channel.

Note: Chaining is also possible to some degree in direct mode, as long as program-mode syntax is used, and no standard BASIC commands are intermixed with JiffyDOS commands.

String Variables

With JiffyDOS, string variables may be used as part of the standard DOS Wedge and JiffyDOS commands. This makes it possible to input disk command strings from the keyboard into a string variable, and then use that variable within a command sent to the disk drive. This makes it easy to design short, memory-efficient, interactive disk "housekeeping" programs with JiffyDOS - something which is not possible with the Commodore DOS Wedge, which forces filenames to be hard-coded into your program.

Example program using string variables:

```
10 REM SCRATCH A FILE
20 @"$"                    Display the directory
30 INPUT"FILENAME";N$      Get filename
40 @"I:"                   Initialize the disk drive
50 @"S:"+N$                Scratch the file
60 END
```

Default Device Override

In JiffyDOS, the current default device may be overridden by specifying a device number at the end of each command. There are two requirements for doing this: First, the command/filename string must be in quotes (as in program mode); and second, the device number must be preceded by a comma. Using the default override is convenient in multiple-drive systems where it eliminates the bother of having to switch the default device assignment back and forth (using the @# or *<CONTROL>+<D>* commands) each time a command is sent to the secondary drive. The default drive override can be used in both direct and program modes.

Examples:

```
@"S0:filename",9      Scratch a file on device 9
@"$:*",9              Display directory of device 9
@"",10                Display device 10 error channel
```

The device number may also be assigned to a variable as in:

```
10 A=9                A=device number
20 @"$",A             Directory of device 9
```


Addendum

The Super Graphics Gold Printer Interface

The Xetec Super Graphics Gold (SGG) printer interface introduces a few quirks into the system when using certain hardware and software. For instance, when using the JiffyDOS device toggle (<CONTROL><D>) with an SGG interface attached, you will see device number 14 show up. This is because the SGG responds to that device number for burst commands. Do not attempt to send JiffyDOS commands to the SGG via device number 14 or a serial bus locckup will occur. This interface can also cause serial communication problems when using a 1571 equipped with JiffyDOS, and serial bus lockups seem to occur much more frequently. We recommend sending the following commands immediately after turning on your equipment when using the SGG interface on a JiffyDOS and 1571 equipped system:

```
OPEN15,4,15:PRINT#15,"SL":CLOSE15
```

These commands should also be sent after pressing the RESET and CLEAR buttons on the SGG at the same time.

Using 1581 Partitions (subdirectories)

You may use the JiffyDOS wedge to move between partitions (often called subdirectories) on the 1581 disk drive. The partition you wish to move to must be visible in the current directory on your 1581, or else you will have to move back to the root directory. Here is the JiffyDOS syntax for moving to a 1581 partition:

Direct: `@/0:name`

Program: `@"/0:name"[,dev#]`

In order to move back to the root directory use the same command without a drive number and name:

Direct: `@/`

Program: `@"/"[,dev#]`

Also note that if you wish to use the JiffyDOS file copier to copy files to or from a 1581 partition, you must use the above commands to change to the proper partition first. For more information on how to create and use 1581 partitions, see Chapter 6 of the Commodore 1581 Users Guide.

Command Summary

Standard DOS 5.1 Wedge Commands

`@`	Read the disk drive error channel
`@C:newfile=file`	Copy a file on the same diskette
`@I`	Initialize the disk drive
`@N:diskname,ID`	Format (NEW) a diskette
`@N:diskname`	Short NEW
`@Q`	Disable the JiffyDOS commands
`@R:newname=oldname`	Rename a file
`@S:file1[,file2]...`	Scratch a file (or files)
`@UJ`	Reset the disk drive
`@V`	Validate a disk
`@$`	Display the disk directory
`@#device`	Set the default device number
`/filename`	Load a BASIC program
`↑filename`	Load and run a BASIC program
`%filename`	Load an ML program
`←filename`	Save a BASIC program

Additional JiffyDOS Commands

`@B`	Disable the 1541 head rattle
`@D:filename`	List a BASIC program from disk
`@F`	Disable the function keys
`@G`	Set interleave gapsize
`@L:filename`	Lock/Unlock a file
`@O`	Un-NEW a BASIC program
`@P`	Toggle printer output
`@T:filename`	List an ASCII file from disk
`@X`	Set Destination device number
`*"filename"type`	Copy a file
`£filename`	Load and run an ML file
`'filename`	Verify a file
`<CONTROL>+<A>`	Toggle all files for copy
`<CONTROL>+<D>`	Default drive toggle
`<CONTROL>+<P>`	Screen dump
`<CONTROL>+<W>`	Toggle single file for copy
`<SHIFT>+<RUN/STOP>`	Load & run first program on disk

`SYS 58451`	Re-enable the JiffyDOS commands (64 mode)
`SYS 58551`	Re-enable the JiffyDOS function keys and commands (64 mode)
`SYS 65137`	Re-enable the JiffyDOS commands (128 mode)

LIMITED WARRANTY

Performance Peripherals, Inc. warrants to the original retail purchaser of the RAMDrive that it is free of defects in material and workmanship for a period of 90 days from date of purchase from an authorized CMD dealer or 90 days from the date of delivery if purchased direct from CMD.

IMPLIED WARRANTIES, OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR OTHERWISE, ARE LIMITED IN DURATION TO THE DURATION OF THE EXPRESS WARRANTY SET FORTH ABOVE. IN NO EVENT SHALL PPI BE LIABLE FOR ANY LOSS, INCONVENIENCE, OR DAMAGE WHETHER DIRECT, INCIDENTAL, CONSEQUENTIAL OR OTHERWISE RESULTING FROM BREACH OF ANY EXPRESS OR IMPLIED WARRANTY, OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR OTHERWISE, WITH RESPECT TO THE EQUIPMENT, EXCEPT AS SET FORTH HEREIN.

SOME STATES DO NOT ALLOW THE LIMITATIONS ON THE LIFE OF AN IMPLIED WARRANTY. SOME STATES MAY ALSO DISALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT PERTAIN TO YOU.

DESCRIPTION OF WARRANTY RIGHTS

From the date of purchase or date of delivery, in the case of a direct sale through CMD, through the warranty period, PPI will, at its discretion, repair or replace any part deemed to be defective at no charge for parts/labor to the original retail customer. During the applicable warranty period wherein PPI will repair or replace defective parts without charge for labor, all warranty inspections and repairs must be performed at a PPI authorized service agency or by PPI itself.

CONDITIONS TO WARRANTY SERVICE

For this warranty to become effective the following requirements must be met:

1. Any postage, insurance and shipping charges of warranted items to a PPI authorized service agency or PPI itself must be prepaid by the original retail purchaser and these costs are not included under this warranty. The original retail purchaser will be charged for shipping, insurance and any other charges related to the return shipping of the item.
2. The dealer's original bill of sale or a charge or credit or delivery receipt must be retained by the original retail purchaser as proof of purchase date of the warranted item and must be presented to the PPI authorized service agency or PPI itself when warranty claims are advanced.
3. The warranty registration card must be filled out and returned to CMD within 30 days of purchase. If CMD does not receive, in good condition, the warranty registration card within the 30 day period, all warranty services are forfeited by the original retail purchaser.
4. Any PPI product being returned for warranty repairs must be in its original shipping container or one of equivalent structure.

EXCLUSIONS FROM THE WARRANTY

This warranty does not cover the specific items/or conditions described below:

1. Equipment which has been damaged due to:
   - Accident, misuse, abuse, fire, flood, or "Acts of God" or other contingencies beyond the control of PPI.
   - Use of incorrect line voltages.
   - Improper or insufficient ventilation.
   - Failure to follow PPI's operating instructions.
   - Improper or unauthorized repair's.
   - Any unauthorized modification to the device.
   - Improper return packaging or damages caused by failure to insure.
2. Damage to warranted items sustained in shipment to the original retail purchaser.
3. Power transformer voltage or Power Supply conversion to foreign or domestic voltage or current frequency.
4. Any damage resulting from the infection of the unit by a computer virus.
5. Routine adjustments.
6. Damage resulting from the commercial use of this unit.

PPI will not be responsible for labor charges of unauthorized service agencies. PPI will not be responsible for labor charges from PPI authorized service agencies or PPI itself except during the warranty period applicable thereto. PPI will not be responsible for the loss or damage to equipment while in the possession of a PPI authorized service agency. PPI reserves the right to make changes in its design and improvements upon its product without assuming the obligation to install such changes on any of its products previously manufactured.

This warranty gives you specific legal rights and you may also have other rights which vary from state to state.

RETURN POLICY

This unit may be returned to Creative Micro Designs, Inc. within 30 days of purchase for a refund of the purchase price less a 10% restocking fee. Shipping charges and taxes are not refundable.

Goods being returned must be returned in original condition in the original shipping container, freight prepaid, and must also include all accessories and be accompanied by a letter stating the reason for return. This letter should contain a return authorization number obtained from Creative Micro Designs, Inc. The return authorization number should also be clearly visible in large characters on the shipping carton.

Problem Reporting Log

If a problem with using this device should occur, or if you need advice or wish to inform PPI of any findings you have discoverd about the unit or its use with specific software packages, please complete a copy of this form and mail to Creative Micro Designs, Inc., P.O. Box 646, E. Longmeadow, MA 01028. When reporting problems with specific software, send a copy if at all possible.

Name: _____

Address: _____

City: _____ State: ____ Zip Code: _____

Telephone #: _____

CMD Device: _____ Serial #: _____

Computer and Peripherals: _____

_____

Detailed Description of Condition: _____

_____

_____

_____

_____

_____

_____

_____

Instructions to Reproduce Condition: _____

_____

_____

_____

_____

_____

_____

_____

_____

_____

(If more room is required, attach a copy of this form to an additional piece of paper.)