C64

Survival Guide to the 1541 Disk Drive

AT-A-GLANCE-REFERENCE

[RETURN] means press the RETURN key...

FORMAT A DISK:

```
type OPEN 15,8,15                                          [RETURN]
     PRINT#15,"NO:yourdiskname,2 digit id"                 [RETURN]
     CLOSE 15                                              [RETURN]
or
     OPEN 15,8,15:PRINT#15, "NO:yourdiskname,id":CLOSE 15
     [RETURN]
```

Wait for light—red light—to go out.)

SCRATCH A FILE/PROGRAM:

```
type  OPEN 15,8,15                                         [RETURN]
      PRINT#15, "SO:name"                                  [RETURN]
      CLOSE 15                                             [RETURN]
or
      OPEN 15,8,15:PRINT#15, "SO:name":CLOSE 15            [RETURN]
```

INITIALIZE YOUR DRIVE:

```
type  OPEN 15,8,15                                         [RETURN]
      PRINT#15, "I"                                        [RETURN]
      CLOSE 15                                             [RETURN]
or
      OPEN 15,8,15:PRINT#15,"I":CLOSE 15                   [RETURN]
```

VALIDATE YOUR DISK:

(never use with a disk with Random files)

```
type  OPEN 15,8,15                                         [RETURN]
      PRINT #15, "V"                                       [RETURN]
      CLOSE 15                                             [RETURN]
or
      OPEN 15,8,15:PRINT#15,"V":CLOSE 15                   [RETURN]
```

READ THE ERROR CHANNEL:

```
type  10 OPEN 15,8,15                                      [RETURN]
      20 INPUT #15,A,A$,B,C                                [RETURN]
      30 PRINT A,A$,B,C                                    [RETURN]
type  RUN                                                  [RETURN]
```

A SURVIVAL GUIDE TO THE 1541 DISK DRIVE

By
MINDY SKELTON

Published by: *MegaSoft, Ltd.*
P.O. Box 1080, Battle Ground, WA 98604

TABLE OF CONTENTS

Introduction 1

Chapter 1: *Setting Up* 3

Chapter 2: *Technical Junk* 5

Chapter 3: *The First Steps* 8

1- Turning it on 8
2- The Disk and how to use it 8
3- Ready-Made Disk 9
(a) LOADing a directory 9
(b) LOADing a file 10
(c) VERIFYing a file 10
(d) LISTing a PRG file 10
(e) RUNing a PRG file 10
4- Your own Disk 11
(a) FORMATting a Disk (or the first time) 11
(b) LOADing a directory 11
(c) LOADing a file 12
(d) VERIFYing a file 12
(e) LISTing a PRG file 12
(f) RUNning a PRG file 13
(g) SAVEing a PRG file 13
(h) SAVE and replace (SAVE@0) 13

Chapter 4: *Second Steps*

Section 1:
(a) SCRATCHing (erase) a file 14
(b) RENAME a file 14
(c) COPY a file 15
(d) MERGE two or more files 15

Section 2:
(a) VALIDATE a Disk 15
(b) INITIALIZE a file 16
Section 3:
(a) Reading the Error Channel 16
(b) OPENing a Channel to the Printer 17
Section 4:
(a) Wild Card or Pattern Matching 17
(b) Making a Backup 18
(c) Disk to Disk COPY 18
(d) Tape to Disk COPY 18

Chapter 5: ***The Wedge and How to Use It*** 19

Chapter 6: ***Files*** 22

Chapter 7: ***Troubleshooting*** 24

Chpater 8: ***Additional Readings*** 26

INDEX 29

SPECIAL NOTE for reading this guide: there is a slight difference between a ZERO and an "O" in the text. However, it is of great importance that you can recognize the difference (because this is not a dot-matrix printer there can be no line through the zero).

A zero is just a bit thinner: 0

An "O" is much more round: O

It is important that you study the difference in the two because you will need to know the difference as you go through the programs and commands listed in the guide...

INTRODUCTION

Congratulations! You now own a Commodore and a 1541 Disk Drive. You've got a great combination. Unfortunately, you've also got the *VIC-1541 User's Manual* and you may be finding it tough to dig out the information you need. That's what this *Survival Guide* is here for. Here you can find the information you need, in plain English, to make the most of your system.

You might want to glance through the entire *Guide*, just to get an idea of what's here and where it is. Don't feel that you have to learn *everything* at once! The *Guide's* broken down into easy-to-swallow pieces. You say all you want to do is load and run your new game disk *Giant Space Marshmallows Attack Harrisburg!*? Fine. Look in the *First Steps* - 'Ready-made disks'. You want to save the nifty little program you just ran off? That's *First Steps* - 'Your Own Disk'. The stupid little red light is blinking and you don't have *any* idea why? Look in *Second Steps* - 'The Error Channel or That Stupid Red Light is On AGAIN!'. It's all here. Relax.

Another purpose for the *Guide* was to correct a few little mistakes in the *User's Manual*. I'll give references to the *User's Manual* so you'll be able to make comparisons.

There's a section here on the *Wedge* program you got with your disk drive. I'll give you some reasons why you'll want to use it, as well as a more comprehensive list of commands.

You can also find some general information on files. I've included a section on very general trouble-shooting and a bibliography of helpful books and articles.

Last but certainly not least is the at-a-glance reference listings on the inside of the cover. You can use the listings found there to jog your memory of just what to type to Format, or Initialize or Read the Error Channel.

Have fun!

SETTING UP

O.K. You've unpacked your disk drive and you're ready to hook everything up. Confident!? If you have no questions, qualms, or if everything is already hooked up an running properly, move right along to the next chapter. For the rest of you, here we go.

Turn off all the power to your system and make sure the on-off switch on the back of your drive is in the "off" position. You should *never* plug anything into your drive when the power is on. Now look at Fig. 1. Plug one end of your serial bus cable into one of the serial bus inputs (it doesn't matter which one you use). If you have a printer or additional drive, you can plug it into the other serial bus. Now plug the other end of your cable into the input on the back of your computer (in the outlet closest to the user port). Plug your power cable (the thing that has an electrical plug on it) into the power input (it will only go in a grounded recepticle) or your power strip.

You've done it! Now you can move on to the next chapter.

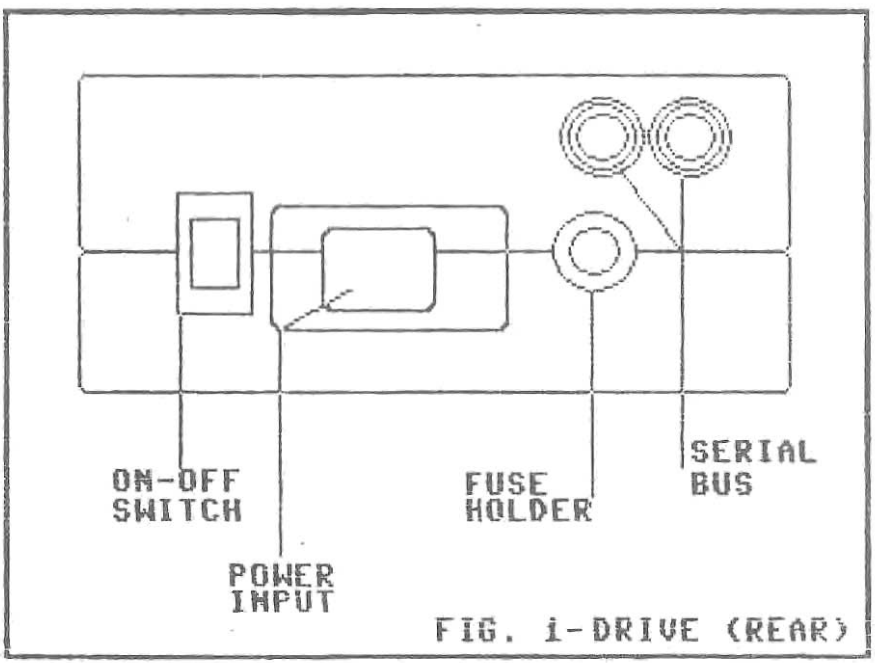

FIG. 1-DRIVE (REAR)

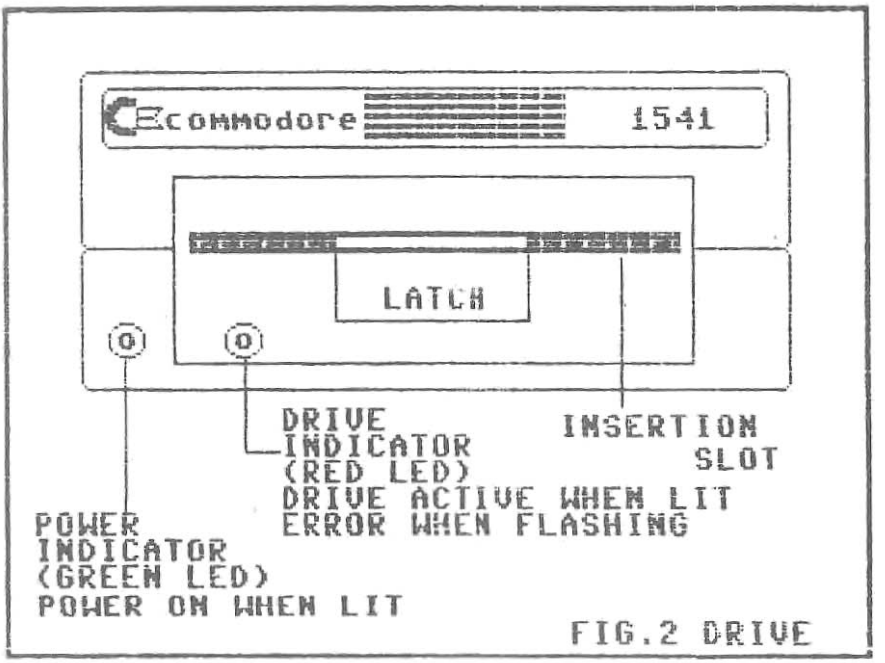

FIG.2 DRIVE

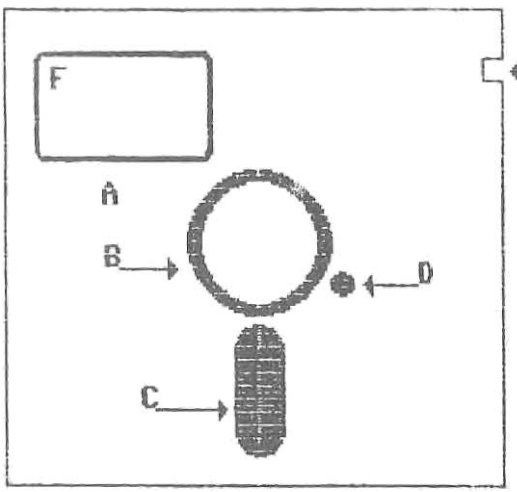

FIG.2.1-THE DISK

KEY:

A-PROTECTIVE COVER
B-HUB
C-READ/WRITE SLOT
D-INDEX HOLE
E-WRITE PROTECT NOTCH
F-PERMANENT LABEL (OPTIONAL)

TECHNICAL JUNK

One of my pet theories about programmers is the *intuitive* vs. *structure freak* division. Intuitive programmers often don't know or care how their programs work, they just do it. Structure freaks, on the other hand, love rules and regulations and knowing *exactly* how and why something works. Users of this guide may also fall into these categories. At the extreme end, if you *never* want to know how or why your drive works, remember this — IT'S ALL MAGIC —; now go on and memorize the commands in the next chapter. If you are a moderate member of the intuitive school, this chapter is still probably not for you, but you may want to look at the pictures. One the other hand, if you are not comfortable without a grasp of the whys and hows of your computer - read on for at least a minimal grasp of what's going on.

The Drive Itself:

Your 1541 disk drive is a device for storing information, in magnetic form, on a disk (in this case, a 5¼ inch floppy disk). The information is stored via a *read/write head.* The read/write head is analogous to the play/record head in a tape player.

When you put a disk in your drive and close the door, the *drive hub* clamps down on the center (hub in Fig. 2.1) to hold the disk in place and center it. The centering is not perfect, so there is some malformation of the disk. This is one reason disk with *hub rings* or reinforcement of the hubs are desirable. The hub ring helps protect the shape of the disk. A pressure pad presses down on the disk from above, insuring proper contact with the read/write head *underneath* the disk. Remember when you are handling your disk that the information is stored on the underside. The head is mounted on a track which permits it to move back and forth across the disk as the disk revolves at approximately 300 RPM.

When you read from or write to a disk, the head moves across the disk to the proper location on the disk. The head comes so close to the disk when the drive is in operation, that even the oil deposited by a fingerprint is thicker than the clearance between the head and the disk.

You may hear from people who sell head cleaning kits that you should use their product once a week. Don't you believe it! Using the kind of cleaner which uses pads had been likened to cleaning your read/write head with sandpaper. Imagine the damage a weekly sandpapering session would do! Cleaning kits relying on strong solvents can soak your pressure pad with solvent and eventually *dissolve* parts of your disk. Great! A once a year cleaning with alcohol should suffice, as should a word to the wise.

The Disk and Disk Structure:

Your disk is a piece of flexible magnetic tape, much like the tape used in a tape recorder. It is contained in a protective covering to prevent damage. The read/write head of your drive accesses the disk through the *read/write slot* (see Fig. 2.1). The density of the magnetic coating determines whether

the disk is *single density* (thinner coating) or *double density* (thicker coating). The strength of the magnetic signal determines whether you need single or double density. Single should suffice for Commodore. If the coating is too thick for the signal, the data may not be recorded properly, and you may lose data.

The disk is divided into *tracks*. Unlike a record which has only one continuous track, a disk has many individual tracks. Commodore Disk Operating System (DOS) uses 35 tracks. Track 1 is the outermost track, while track 35 is the nearest to the hub. Track 18 holds the Directory (q.v.) and the Block Allocation Map (BAM q.v.). Each track is further divided into *sectors* or *blocks*. The Commodore is able to store 170K (one hundred and seventy thousand bytes) on a single density (q.v.) disk by varying the number of sectors in each track. Tracks 1 to 17 hold 21 sectors each. Tracks 18 to 24 hold 19 sectors each. Tracks 25 to 30 hold 18 sectors each, while tracks 31 to 35 hold 17 sectors each.

The BAM:

The letters in BAM are variously given as representing *Block Allocation Map, Block Availability Map* and *Bit Access Map*. Whatever you decide it stands for, the BAM is a list of all the blocks/sectors on your disk (683 in all). Each block holds 256 bytes. (If you multiply 683 by 256 you get 174,848 bytes available, or about 170K.) The BAM keeps track of whether a given block is in use or not by storing a block of bits which correspond to the available location on your disk. By turning the bits on or off (1-on/available or 0-off/unavailable) the BAM keeps track of your available space. The BAM is updated each time a file is SAVEd or SCRATCHed, or every time a data file is *closed* (note difference in directory storage method).

The DOS stores the BAM in its memory and goes on its way. If you change disks, the DOS is still operating on the stored BAM so that if you try to write to the new disk, blocks that the BAM says are available, may not be. The DOS checks ID numbers before it writes to the disk, so if you have unique ID numbers for all your disk, the worst thing that will happen is that you get an error. If, however, you use the same ID number on all, or many of your disks, the DOS will assume everthing is alright and you could end up overwriting your files. Unique ID numbers are a must (see *What Happens When you FORMAT).*

Even keeping this in mind, it's always a good idea to initialize your drive (q.v.) whenever you change disks.

Directory Facts:

The directory is stored in track 18 of your disk. It contains the disk *header* (the disk Name and ID number) stored in block 0. The rest of track 18 contains the names of all the files on the disk, the file designation for each (PRG, SEQ, REL, USR), the number of blocks used by each file and the number of free blocks on the disk. You can store up to 144 file entries in your directory. The directory is updated every time a program is SAVEd or SCRATCHed, or when a data file is opened (note difference in BAM update methods, below).

What Happens When You FORMAT:

When you FORMAT (or as it is sometimes called NEWing) a disk, you erase everything that was on the disk (if it was previously used) and get it ready to be written to and read from. The Disk Name is stored in track 18, block 0, and the *unique* ID number is written to every block on the disk. The directory is begun in track 18. Immediately after the disk is FORMATted, the directory consists of the Disk Name, the ID number, the ASCII code for 2A (the DOS version and format type) and the number of blocks free (664 - the 19 blocks of track 18 are set aside for the header and directory).

What Happens When You SCRATCH Files/Disks:

When you SCRATCH a file the bit in the directory which indicates the file type is set to 0 (deleted). The File Name is no longer listed in a directory listing, and the blocks allocated to that file are considered free for other use. The file is actually still there, and if the relevant bit can be reset to the appropriate number (1 - 4 depending on file type) the file can be restored.

Resurrecting Files/Disks:

Files:

A file can be *brought back* either by a *canned* program specially designed to *unscratch* a file, or by gaining access to your disk's BAM and resetting the bit that designates whether a file is determined to be *present.* When a file is SCRATCHed this bit is set to 0. Setting it to 1 will bring back the file, unless you have used the disk and overwritten parts of the file.

Disks:

Forget it! Once you NEW or reFORMAT a disk, all the files that were on that disk are gone. If anyone out there knows of a way to bring them back, I would love to know about it. (By the way, if you can do that, why are you reading this book!?)

THE FIRST STEPS

1- Turning It On

First, make sure there is no disk in your drive. You should *never* turn your drive on or off with a disk in it. You could ruin the disk. Once you're sure the drive is empty, you are ready to turn on the power. The argument concerning the order in which you turn your components on and off seems to be the modern equivalent of how many angels can dance on the head of a pin.

One school (including Commodore) holds the computer should *always* be turned on last and off last; another school holds the computer should be turned on first and off first. There are other opinions which incorporate both. Check for any special instructions that your printer or interface might have. If you have all your components plugged into a power strip or surge suppressor, everything will go on and off at the same time and saves you the effort of worrying. I personally lean toward the *computer on last off last* school, but I can give you no *ex-cathedra* pronouncement on this subject.

When you turn on the drive, both the green light and the red light will come on. [See *Technical Junk* Fig. 2] The red one should go off almost at once. The green light stays on all the time the drive is on. The red one is only on while the drive is running. *NEVER* take a disk out while the red light is on and the motor is running. You may want to make a correction in your Commodore *1541 Users Manual,* page 8. In the last secton of *Insertion of the Diskette*, cross out *green* and insert *red.* If you wait for the green light to go out, you may have a very long wait. *(Don't read the next sentence unless the red light is blinking and you HAVE to remove the disk for some reason.)* Actually, it's a good idea not to take a disk out of the drive while the red light is on, but if the red light is blinking to indicate an error, and the motor is NOT running, you can take the disk out if you must.

2- The Disk and How To Use It

Your *User's Manual* specifies single-sided, single density disks. Most disk manufacturers recommend single-sided, double density. You can probably use either. [See the section *Technical Junk* for more on density.]

Disks don't respond well to very high or low temperatures. As a rule of thumb, if you're comfortable, so is your disk. When you transport a disk, or buy a disk, or get a disk in the mail, it's a good idea to give the disk a chance to *rest* and come to room temperature before you try to use it. A disk that won't operate properly when it's cold, may work just fine after it's more comfortable. Never leave disks in direct sun or in the glove compartment of your car.

You should also avoid taking *floppy* too literally. Try not to bend the disk as this can cause the magnetic coating to come off in spots.

Also try to avoid magnetic fields (such as those generated by mail sorting devices and MONITORS and TVs). If you must send a disk through the mail, mark it for *hand cancelling.* Never store a disk on your monitor, and if fact try not to store them closer than two feet away.

Never write on a disk with anything other than a felt tip pen. Anything else can leave marks on the magnetic surface of the disk.

Once you have your disk, remove it gently from its protective sleeve, holding it by the upper left corner (as you look at it). [See *Technical Junk* Fig. 2.1 for more info.] There is a notch in the right side. This is the *Write Protect* notch. If there is tape over this notch you cannot write (save) anything on the disk. A small digression: You should never actually *write* anything on your disk. Use felt tip pens to mark on your label. Now, still holding the upper left corner, gently insert the disk, with the label up, in the insertion slot on the front of your disk drive (which is *already* turned on). [See *Technical Junk* Fig. 2.] *NEVER force a disk. Once the disk is in place, shut the disk drive by pulling down on the latch until it clicks shut, or in newer models, turning a latch. Fine, now you're ready to go.*

In the following sections the word file will be used where you expect the word *program.* This is because a program is a kind of file.

3- A Ready-Made Disk

By *ready-made* or *used disk,* I mean a pre-packaged disk that you have purchased (a game or a data base or the like)—one that is already formatted and has something on it.

(a) LOADing a directory

This means that you want to put into your computer's memory a lising of all the files available on a disk. Once you see the directory, you can decide what file to put in memory. You may notice that each of the file names has a number in front of it and a three-letter code following it. The numbers are the number of blocks taken up by the file, and the codes indicate what kind of file it is. The codes are PRG (program file), SEQ (sequential file), USR (user file - identical to SEQ file), and REL (relative file). Programs (PRG) are the only files which can be loaded directly, so in the following sections, when you see *file*, it's probably alright to think *program.* The other types of files (SEQ, REL, and USR) are all *data* files (they hold data, not an actual program) and can only be loaded through a PRG file. We'll talk more about this later. Some pre-packaged disks have loading instructions so that it's not necessary for you to see the directory. Some disks are even protected so that you *can't* see the directory. But assuming both that you want to and can, *here's* how you do it. Type:

LOAD"$",8 [RETURN]

Your computer will respond with:

```
SEARCHING FOR "$"
LOADING
READY
```

If you see something different, you have a problem. (See the *Trouble-Shooting* section). Once you see "READY" you can type "LIST", "RETURN", and your directory will be printed on your screen.

(b) LOADing a file

This means entering a file into your machine's memory so that you can use it. This is what you do. First you choose which file you wish to load (hereafter referred to as *name of your file* - remember you type the actual name of the file!). Then type:

LOAD"name of your file",8 [RETURN]

Your computer will respond with:

SEARCHING FOR "name of your file"
LOADING
READY

If you see something different, you have a problem. (See the *Trouble-Shooting* section). Once you see "READY" you can do three things (actually you can do many more, but let's not get philosophical): you can VERIFY the file, LIST the file or you can RUN the file.

(c) Verifying a file

When you verify a file you've just loaded, you compare the file in memory to the file on your disk. If they are not identical, you will see VERIFY ERROR and you will know something is wrong. To verify a load, type:

VERIFY"name of your file",8 [RETURN]

The computer will respond with:

SEARCHING FOR name of your file
VERIFYING
OK
READY

(d) LISTing a PRG file

If you want to see a listing of the PRG file, and the file will let you (and for a number of reasons a ready-made file may not) you can type the command:

LIST [RETURN]

The lines of your file will now scroll past you on the screen. We'll talk more about LIST in the next section.

(e) RUNning a PRG file

If everything has gone well up to now, you are ready for the big plunge. Running your file! When you give this command the computer looks at whatever is in its memory (and you've just put your file there) and follows the commands of the file. So you type:

RUN [RETURN]

and you're off and running. (You are, aren't you?)

4- Your Own Disk

Now we're talking about your very own disk. Maybe it's right out of the box and has never been used. Maybe it's got some of your most precious files on it. Let's see what we can do with it. (Please read Section (a). It's *really* important!)

(a) FORMATting a disk (or The First Time)

(Additional information on the process of FORMATting and the disk itself can be found in the section *Technical Junk*.)

When you take that nice new disk out of the box, it's not ready to be used. Before a disk can be used, you must *FORMAT* it. When you FORMAT a disk you give it a name and identification number. The identification number can be any two characters and should be unique for each disk. (You might want to use ID numbers that reflect what is on the disk. You could start all the IDs of game disks with a G, or all utility disks with a U, etc.) The Identification number (ID #) is *written* on each sector of your disk. The formatting process also sets up a directory for your disk. Once there is a directory and each sector has been identified, your disk is ready to use. To format your disk, type:

```
OPEN 15,8,15                        [RETURN]
PRINT#15,"NO:DISK NAME, ID#"        [RETURN]
CLOSE 15                            [RETURN]
```

The red light will go on and your drive will start to make noises. The READY prompt will appear on your screen, but the red light will stay on. The disk is not formatted until *the red light goes out*. Your disk should now be ready to use. You only need to format a disk the first time you use it. If you re-format a disk, everything that was on it before will be lost, so *never* re-format a disk unless you want to erase it.

Well, what can you do with your formatted disk? The first thing you might want to do is *LOAD the directory* to make sure everything is alright.

(b) LOADing a directory

Loading a directory means that you put into the computer's memory the *name* and *ID#*of your disk and a list of all the files (if any) on the disk. Whether this is an empty disk, or one that you have filled with files, the process for loading a directory is the same. You type:

```
LOAD"$",8  [RETURN]
```

Your computer will respond with:

```
SEARCHING FOR "$"
LOADING
READY
```

If you see something different, you have a problem. (See the *Trouble-Shooting* section.) Once you see READY you can type LIST, RETURN, and your directory will be printed on your screen.

You can now choose a PRG file from the directory to load into memory.

(c) LOADing a file

LOADing a file means getting a file from some storage device (in this case, a disk) and transferring the instructions into your machine's memory so that you can use it. Here is what you do. You have already chosen your file (see section 3-b). Hereafter we will refer to it as *name of your file* although you type the actual name of your file. Then type:

LOAD"name of your file",8 [RETURN]

Your computer will respond with:

SEARCHING FOR "name of your file"
LOADING
READY

If you see something different, you have a problem. (See the *Trouble-Shooting* section.) Once you see READY you have some choices: you can VERIFY the file, LIST the file or you can RUN the file. Let's see what those choices mean.

(d) VERIFYing a file

When you VERIFY a file you've just loaded, you compare the file in memory to the file on your disk. If they are not identical, you will see *VERIFY ERROR* and you will know something is wrong. To verify a load, type:

VERIFY"name of your file",8 [RETURN]

The computer will respond with:

SEARCHING FOR name of your file
VERIFYING
OK
READY

(e) LISTing a file

If you want to see a LISTing of the PRG file currently in memory, you can type the command:

LIST [RETURN]

The lines of your file will now *scroll* past you on the screen. To slow this scrolling, press the control key. (That's the key marked CTRL.) If you want to see only part of a program you can request sections of the program to be listed. If you want to see lines 10-50 (for example) type:

LIST 10-50

If you want to see all the lines up to line 100 (for example) type:

LIST -100

If you want to see all the lines after 100 (again, as an example) type:

LIST 100-

You can then make corrections, check syntax or whatever you wish.

(f) RUNning the PRG file

If everything has gone will up to now you are ready for the big plunge. Run your file! When you give this command, the computer looks at whatever is in its memory (and you've just put your file there) and follows the commands of the file. So you type:

RUN [RETURN]

and your program begins! There, wasn't that easy!?

(g) SAVEing a file

If you have entered a program, either by hand or from some other storage source (e.g., disk or tape) and wish to keep a copy on the disk now in your drive, you can use the *SAVE* command. This command makes a copy of the program in your machine's memory onto the disk in your drive. You decide what you want to call the file (in our example we'll use "your file name" but you use the actual name) and then type:

SAVE"your file name",8

Your 64 will respond with:

SAVING your file name
READY

You can't save two programs with the same name on the same disk. If you want to save multiple copies of the same program, you must give the files different names (e.g. PROGRAM1, PROGRAM2, etc.).

You can VERIFY a save just as you would a load by following the instructions given before. In this case the verify checks to see if the program has been saved as written.

(h) SAVE and REPLACE (SAVE@0)

The command SAVE@0 replaces a file with a new version (with the same name) by making a copy of the new file, and then SCRATCHing the old file. Your Commodore User's Manual says you can use a command SAVE@0: to save and replace an already existing file with the same name. Yes, you can, but...

There is an ongoing debate in the Commodore community concerning this command. One school holds that you run the risk of losing one or both files, or in extreme cases, the entire disk if you use SAVE@0. There is, reputedly, a bug in this command which makes it risky to use and sometimes causes it to replace the wrong file. If you want to follow this school, scratch your file and resave it. (Look in the next chapter under *SCRATCHing a file* for help.)

Other Commodore experts swear that SAVE@0 is completely safe, and that problems allegedly stemming from its use can always be traced to other causes. Even this school admits that SAVE@0 cannot be safely used if there is not enough room on the disk for both the original and the replacement to exist at the same time (which is what happens before the first version is erased). That is, if you only have 10 blocks free, don't try to use SAVE@0 with a file that is 20 blocks long.

When you feel comfortable with the commands in this secton, give a look at the next chapter - *Second Steps.*

SECOND STEPS

Now that you have the basic moves down, you can think about some of the more advanced things that you and your disk drive can do. In this chapter we will look at ways to send your programs to a printer, erase a file, read your error channel and more. As in the last chapter, the name "your file name" will be used in the examples, where you would type the actual file name. First let's look at how to remove an unwanted file.

Section 1:

(a) SCRATCHing (erase) a file

SCRATCHing a file, in effect, erases a file from your disk, although in fact what it does is erase the name of the file from your directory. (See *Technical Junk* for more on how the directory works.) When a file is scratched, the number of blocks free is increased by the number of blocks that were in the file. When you see [RETURN] press the key marked RETURN. In order to scratch a file, type:

```
OPEN 15,8,15                          [RETURN]
PRINT#15,"SO:your file name"          [RETURN]
CLOSE 15                              [RETURN]
```

The drive will make some noises and the red light will come on while the program name is erased. Be careful when you scratch a file, as it is *very difficult* (often impossible) to restore a scratched file.

If you wish to save a file and replace an already existing file with the same name, SCRATCH the file on your disk using the procedure given above, then save the new file. This will avoid the problem with the SAVE@0: command (q.v.).

In the following sections we will look at ways to change the name of an already existing file (RENAME) make a copy of an already existing file (COPY) or combine two to four files (MERGE).

(b) RENAME a file

If you wish to change the name of a file on your disk, this is the command for you. One word of warning: This command does not work on an open file, so wait until you close your file to rename it. In our example the terms *oldname* will stand for the current file name and *newname* will stand for the new name you wish to give the file. Now, taking all that into consideration, this is how you *RENAME* a file. Type:

```
OPEN15,8,15                           [RETURN]
PRINT#15,"RO:newname=0:oldname"       [RETURN]
CLOSE 15                              [RETURN]
```

(c) COPY a file

This command will make a copy of a file on your disk. It will not copy from one drive to another unless you are using a dual drive (such as a 4040 or MSD dual drive). In our example, *newfile* will be the name of the copy and *oldfile* the name of the file to be copied. You will, of course, enter the actual names. Type:

```
OPEN15,8,15                                  [RETURN]
PRINT#15,"CO:newfile = 0:oldfile"            [RETURN]
CLOSE 15                                     [RETURN]
```

(d) MERGE two or more files

The COPY procedure can be used to copy two to four files into one new file. When you use this procedure, make sure that the files you are going to merge don't have overlapping line numbers, as this can really make life difficult for you. If you find that there *are* identical line numbers in the programs you wish to merge, renumber one or more of the programs using either a programmer's aid such as VIC Tree or Simon's Basic, or get a copy of one of the Public Domain renumbering utilities from your local user group. Once you have checked the line numbers you can proceed to MERGE the files. Again, our example will use *newfile* as the name of the copy you are creating and *oldfile* (to a maximum of four) as the name of the files to be copied. Type:

```
OPEN15,8,15                                  [RETURN]
PRINT#15,"CO:newfile = 0:oldfile1,oldfile2,
    oldfile3,oldfile4"                       [RETURN]
CLOSE 15                                     [RETURN]
```

Section 2:

The next two commands we will examine don't do anything to a particular file; instead they enable you to *clean up* a disk (Validate) or to return the drive to its *start up* condition (Initialize).

(a) VALIDATE a disk

There are number of reasons you might want to use this command. If you have been using a disk for a long time and have erased a number of files, you will have small groups of blocks here and there which were too small to be of any real use. The VALIDATE command will *clean up* the disk by causing the drive to search the disk for these unused blocks and making them available to you. You may be able to increase the *blocks available* by a considerable amount by VALIDATING. Another time to use VALIDATE is when a file will appear in the directory with an asterisk (*) in front of the file type (PRG, SEQ,USR, REL). Improperly closed files can cause problems with all the information stored on the disk. You cannot SCRATCH such a file without risk, but VALIDATING the disk will remove it. To VALIDATE a disk, type:

```
OPEN15,8,15                                    [RETURN]
PRINT#15,"VALIDATE"                            [RETURN]
   (instead of VALIDATE, you can use PRINT#15,"V")
CLOSE 15                                       [RETURN]
```

(b) INITIALIZE

This command returns your drive to the same condition as when you turned it on. INITIALIZE clears the error channel (this is one way besides reading the error channel (q.v.) to get that stupid red light to stop flashing!), and if used every time you switch disks, will prevent any possible confusion caused by different BAMs (block allocation maps (q.v.) on your different disks. To INITIALIZE your drive, type:

```
OPEN15,8,15                                    [RETURN]
PRINT#15,"INITIALIZE"                          [RETURN]
   (instead of INITIALIZE, you can use PRINT#15,"I")
CLOSE 15                                       [RETURN]
```

Section 3:

Now let's look at some things you can do which involve *opening channels* to other components, either your drive or your printer.

(a) Reading the ERROR CHANNEL, or "That Stupid Red Light is Flashing Again!"

When there is an *error condition*, for example, a read error, or the file you are trying to access is not open or any of a number of problems (see pp. 43-46 of your *User's Guide*), the red light on your drive will start to blink rapidly. Unfortunately, there is no immediate way to know *why* you are getting the error light. What you need to do now is *read the error channel.* This means opening up a channel (channel 15 is reserved for error messages) and *asking* the computer for information regarding the error. You open a channel by using the command OPEN followed by a file number, a device number ('8' for your disk drive, '4' for your printer) and a channel number. Usually one makes the channel number (which can be any number from 2 to 14 - 0, 1, and 15 are reserved) the same as the file number. You will see this structure in the sample program below (look at line 10).

If you are not using your *Wedge* (q.v.) this program will get you the information you need.

```
10 OPEN 15,8,15
20 INPUT#15,A,A$,B,C
30 PRINT A,A$,B,C
40 CLOSE 15
```

The four pieces of information you get are (A) the error number (refer to pp. 43-46 of your *User's Guide* for additional information about the errors), (A$) the error name, (B) the track number, (C) the sector/block number (see *Technical Junk* for further explanation). If you don't care about the track or sector, you can just request the first two pieces of information (A and A$). You now know your problem and can take the appropriate steps to correct

it. Also the red light is off.

As an example, suppose you tried to save a file and suddenly the red light started blinking, indicating a problem. You have no idea what's wrong. You look at your program and see nothing obviously wrong. Here's what you do. Let's say our imaginary program starts with line 10. You then enter an abbreviated form (entering only the first two variables) of the program on p.22, beginning your line numbers with 1 rather than 10. You also add line #5 END. This numbering prevents you from overwriting the program, and the line 5 stops the program as soon as the relevant information has been obtained. You RUN the program and get the information, 63 FILE EXISTS. You would know that you were trying to use the name of an already existing file and you could try and SAVE your program again under a different name.

If you want the red light off, but don't care what the error message is, INITIALIZE your drive. (See previous section for directions.)

(b) Opening a channel to your printer

If you want *hard copy*, or a printed listing of your program, you have to tell the computer to send the output to your printer rather than your screen. The following command will do just that (compare with the incorrect instructions on p. 11 of your *User's Guide*):

OPEN 4,4:CMD4:LIST [RETURN]

To return control to your screen, type:

PRINT#4:CLOSE4 [RETURN]

There is no logical reason you should have to type PRINT#4 before you can close the channel, but trust me, you do.

Section 4:

The last section of this chapter will explain Wild-Card or Pattern matching, how to make a backup copy of your disk, and how to copy files between disks and between disk and tape.

(a) Wild-Card or Pattern Matching

There are two *wild-card* symbols used in matching - '*' and '?'. By using a *Wild-Card* symbol, you can save yourself some typing by having to type only part of a program name to LOAD a program. For example, typing 'LOAD "P*",8' would load the first progam on a disk whose name began with 'P'. Typing 'LOAD "O:*",8' will automatically LOAD the first program on a disk. As long as you give enough of the name to make it unique, you can use pattern matching. You can also use pattern matching to SCRATCH file, but use it with caution so that you don't scratch more than you wanted to. for example, the other night my husband typed 'SCRATCH "*.DAT"'. What he meant to do was scratch all the files ending with a .DAT ending (our way of designating data files); what he did instead, was erase every file on the disk, and it took a *LONG* time to get them back!

The other *wild-card* symbol, '?' can be used within a file name when you are unsure of the entire name, or when you want to list a directory of only certain files. For example 'LOAD "B?LL",8' could load files called "BILL",

"BULL", "BELL", etc.. Typing 'LOAD "$O:F?LL",8' would load a directory of any programs on the disk whose four-letter names began with 'F' and ended with 'LL'. On the other hand, 'LOAD "$O:F?LL*",8' would load a directory of all programs with names of any length whose first letter was 'F' and whose third and fourth letters were 'LL'.

(b) Making a backup

Why would you want to make a backup? Well, disks are not indestructible. They are subject to wear and tear, heat, dirt, spilled cokes, cats and children. There's no excuse for losing a valuable disk when you can make a copy of it yourself.

To make a backup copy of an entire disk, you need a *copy program.* There are several good ones available through public domain channels. Some of the most popular are *1541 Backup* (reliable but takes up to 30 minutes for full disk), *B 24 File Copier* (faster and gives you the option of copying idividual files) and *Four Minute Backup* (just as the name implies, but can be scary to use). There are also copyrighted programs such as *Clone Machine, Supercopy 64* and *The Jolly Roger* which copy even some protected disks (for archival purposes ONLY). There are many more programs both in your favorite store and in your user group library. Get one as soon as you can.

(c) Disk to Disk Copy

To copy a non-machine language program from one disk to another, simply LOAD the program into your machine memory in the regular way (See LOAD), remove the first disk (source), insert another formatted disk (destination) and issue a SAVE command (See SAVE). There, all done!

To copy a machine language program, you will either need a program specifically designed to transfer machine language programs or a program which allows you to specify files to be copied.

(d) Tape to Disk Copy

To copy a non-machine language program from tape to disk, LOAD the program from tape into you machine memory as you normally would, insert a formatted disk and issue a SAVE command [See SAVE]. Reverse the process to save from disk to tape.

THE WEDGE AND HOW TO USE IT

Many of you will have come in contact with a utility referred to as the *Wedge,* or in your 1541 manual p. 14 as the DOS support system, either from your demo disk, your Disk Bonus Pack or your user group (you *are* in a user group aren't you!). This utility loads in *above* BASIC memory (it *wedges* itself in - clever huh?) and lets you use all the commands you are now familiar with: LOAD, SAVE, COPY, FORMAT, etc.. The Wedge gives you a new command - LOAD and RUN.

The Wedge also lets you look at your directory without overwriting the program currently in memory (called a non-destructive load, since it doesn't destroy the program in memory). Another bonus of the Wedge directory is that you can *freeze* the listing by pressing the space bar. Press it again to resume the listing. In addition to your regular directory, you can get partial listings using pattern matching. You can ask for a particular file, or all files which fulfill a particular condition. If, for example, you want a listing of all the files starting with A, enter @$:A. To find, as an example, a file called "sample.dat", enter @$:sample.dat. If you wanted to see if your disk contains, say, the files "sample", "sample1", and "sample2", enter @$:sample*.

Many people don't use their wedge because, surprisingly (or maybe not), the documentation for using it is poor. If you don't have a copy of the wedge, try and get hold of one of the public domain copies.

Once you get your copy, you may want to transfer it to other disks for convenience. When you do this you will note that the Wedge program is in two parts; the *Booter* (which may have *boot* in its name) and the actual Wedge program. On the Disk Bonus Pack these programs are called, respectively, *C-64 Wedge* and *DOS 5.1*. C-64 Wedge is the *booter*, a program written in BASIC which you load and run and which loads the wedge for you. If you don't wish to use the booter program, you can activate the Wedge by entering these lines in direct mode.

```
LOAD "DOS 5.1",8,1          [RETURN]
NEW                         [RETURN]
SYS 52224                   [RETURN]
```

The Wedge is written in machine language (which is why you use the ,8,1 to LOAD it) and is activated by a system (SYS) command (in this case SYS 52224). If you exit the Wedge for some reason, and later wish to reactivate it, you can enter SYS 52224, [RETURN]. This will reactivate the Wedge unless you have turned off your machine or loaded in another machine language program which has interfered with the Wedge.

The fact that the Wedge is a machine language program leads to a problem for you if you want to transfer the wedge to another disk. You cannot copy machine language program simply by LOADing it and SAVEing it to

another disk as you can a BASIC program. You will need a program which allows you to copy individual files from a disk. Such programs are readily available from public domain sources.

So now you have your Wedge and it's ready to run. For your convenience, I will now offer you a handy reference guide on using your wedge. On the next two pages you will find a comparison between BASIC and the Wedge. In one column I will list the syntax of the command as it is in BASIC; in the other column I will give the equivalent using the Wedge.

DOS WEDGE	BASIC
COPY: @C:newfile = oldfile	OPEN 15,8,15 PRINT#15,"CO:newfile = oldfile" CLOSE 15
FORMAT: @N:disk name,id	OPEN 15,8,15 PRINT#15,"NO:disk name,id" CLOSE 15
INITIALIZE: @I	OPEN 15,8,15 PRINT#15,"I" CLOSE 15
LOAD: (non-machine language programs)	
/program name	LOAD "program name",8
(machine language programs)	
%program name	LOAD "program name",8,1
LOAD and RUN program name	LOAD "program name" RUN
READ DIRECTORY: @$	LOAD "$",8 LIST
READ ERROR CHANNEL: @	OPEN 15,8,15 INPUT#15,A,B$,C,D PRINT#15,A,B$,C,D CLOSE 15
RENAME: @R:newname = oldname	OPEN 15,8,15 PRINT#15,"RO:newname = oldname" CLOSE 15

SAVE: program name	SAVE "program name",8
SAVE & REPLACE: (* see note below) @:program name	SAVE "@0:program name",8
SCRATCH: @S:program name	OPEN 15,8,15 PRINT#15,"SO:program name" CLOSE 15

Also note: @Q will terminate the DOS Wedge program.

The symbol can be substituted for @ in any of the Wedge commands.

You can also adjust the speed of your 1541 to accommodate either a C-64 or a VIC-20 with the following command:

For C-64 @UI+
For VIC @UI–

* The SAVE@0 command in the Wedge is subject to the same debate as its regular DOS counterpart. If you have not done so, read about SAVE@0 (pp. 18-19). If you want, you can SCRATCH the file and re-SAVE it in two separate operations, instead of using SAVE@0.

FILES

Since this is a guide to your Disk Drive and not a guide to file programming, the information in this chapter will be somewhat sketchy. Additional information sources on files and file handling are listed in Chapter 8 - 'Additional Reading.'

Your Commodore stores information on your disk in the form of *files*. Files are either *program* files or *data* files. Files are identified by *filenames* of up to 16 characters. No two files on a single disk can have identical names. An example of a program file would be a BASIC program which you have SAVEd to disk. This kind of file can be accessed by the normal DOS commands you have been studying. Data files hold information which is accessed either *sequentially* (as in Random or Relative files). Data files cannot be manipulated by the normal LOAD and SAVE commands, but rather have their own special syntax.

A data file is often likened to a file cabinet, where the file itself is the cabinet, the *records* or entries are the information folders, and the *fields* which hold the information are the papers stored in the folders.

Sequential Files

In a sequential file, data is both read and written in order, that is in sequence. Information is written onto the disk from RAM, through a buffer, one byte at a time, and read back the same way. In order to reach record #7, for example, you must search through records 1-6.

The general syntax for opening a sequential file is:

OPEN file#,device#,channel#,"O:filename,type,direction"

Here's an English translation:

file#	-	the number by which you will refer to the file.
device#	-	8 for disk drive; 1 for tape.
channel#	-	a number from 2 to 14 (remember 0,1, & 15 are reserved). Often the same number as the file# to help prevent confusion.
filename	-	the unique name you assign to the file.
type	-	the kind of file (in this case, sequential).
direction	-	either write (data to disk - can be abbreviated to W) or read (data from disk - can be abbreviated to R).

Information is sent to a sequential file by the PRINT# command. This command operates just like a regular PRINT statement except it sends the information to the drive. Information is separated by what is called a delimator or terminator (a comma, semi-colon or carriage return). Information is retrieved from a sequential file either by the INPUT# or GET# commands.

Random Access Files

The two types of random access files operate differently from sequential files. With random access files it is possible to go directly to the desired record without going through the intervening records, for example, we could go directly to record 7 (and collect $200) without bothering with records 1-6.

Relative Files

Relative files can be thought of as a file whose records are sequential files. When you initially set up a relative file you must specify how long the record is going to be (up to 254 characters). You may place up to 720 records on a disk. The general syntax for setting up a relative file is:

OPEN file#,device#,channel#,"filename,L," + chr$(x)

Again, in English:

file#	-	the number by which you will refer to the file.
device#	-	8 for disk drive; 1 for tape.
channel#	-	a number from 2 to 14 (remember 0,1 & 15 are reserved). Often the same number as the file# to help prevent confusion.
filename	-	the unique name you assign to the file.
,L,	-	indicates a relative file is being created - omit ,L, when accessing a previously created file.
chr$(x)	-	the value of x determines the record length (up to 254) - only specified when initially creating file.

To access an already created file the syntax is:

OPEN file#,device#,channel#,"filename"

To access a relative file you must first open the error channel (q.v.), then tell the DOS which record you wish to access, by specifying the file (by the channel# as given in the OPEN statement), the record within the file and where in the record you wish to begin. (Sigh!! Confused? I told you this would be sketchy. Read some more detailed articles.)

Once you have accessed the file, you can either read from or write to the file with PRINT#, INPUT#, and GET#.

Random Files

Random files are more complicated than either Sequential or Relative files. With Random files you take control of the information away from the DOS. You specify which trach and sector each piece of data will be read from (block read command). You keep track of which blocks are allocated blocks to be used again (block free command). Random files are not included in the disk directory. It is generally not advisable to put anything else on a disk which will be used for Random files. Because of their complexity, I strongly recommend you postpone using Random files to later on in your computing career (if ever).

TROUBLE-SHOOTING

This chapter will offer some suggestions on how to *fix* some problems you may run into. It isn't meant to be a maintenance manual. There are several of those on the market and if you are at all *handy* , I would advise you to get one. Repair costs *average* about $50 per hour, so you can save *big bucks* if you can do a little work yourself. Now - back to *our* helpful hints.

All-Purpose Hint #1

First, when confronted by a problem with your drive, assume it's something fixable and not a total disaster (there's plenty of time for panic later). This means check all your equipment before you start screaming. Be sure your system is all turned on, that the connections are tight, etc.. This amy seem silly, but more than one peron has gotten a *device not present error* simply because the drive was not turned on! If, after you have checked all the connections, you still have a problem, shut off the system and start everything up again. Still have a problem? Don't send your drive back to the store yet. Try it with a friend's computer. If it works with theirs, the problem may be in the computer, not the drive.

All-Purpose Hint #2

Generally, my philosophy of life is "If it's not broken, don't fix it." Sometimes with disk drives, this may not be a good idea. Your drive can and will get out of alignment. This is a gradual process, often so gradual that you may not know anything is wrong. Disks prepared on your own drive may not present problems for you until your drive becomes seriously sick.

If you are starting to have problems with disks not prepared on your drive (ie pre-recorded disks or a friend's disk), but not your own, try realigning your drive. Often this *preventive maintenance* will correct your problem.

Device Not Present Error

(See All-Purpose Hint #1)

Load Errors:

If you are having problems LOADing a particular program after you are sure that everything is as it should be with your equipment, there are several things you can try. Check your syntax. Make sure you gave the LOAD command properly. Try to LOAD another program. If it LOADs properly, the problem may be in the software rather than in the drive. Try to LOAD the program on a friend's machine. If it LOADs properly on another machine, make a backup of it on your own drive (if it's not a protected disk). Sometimes the differences in speed between drives interfere with a program LOADing. For example, I took back three copies of one particular game before I solved

my problems by making a backup on my own drive. I never could get a store copy of that program to work on my drive!

You may also experience LOAD errors if you try to LOAD a file from a disk with an improperly closed file (q.v.). If this is the problem, VALIDATE the disk (q.v.) and try again.

Save Errors

First check to make sure there is no *write protect* tab covering the notch in the upper right-hand corner of your disk. When that notch is covered, you can't SAVE anything to that disk. Make sure there is not already a program on the disk with the same name as the name you are trying to give your program. If you are using the *Wedge*, you can do this by typing "@$"; otherwise, just try to SAVE the program again with a different name (e.g., "programname!"instead of "programname"). Don't check your directory undless you are using the Wedge (or something like it) or you will overwrite your program and lose it. Check your syntax. Be sure you gave the command properly. Check to see if the disk is full (0 blocks free). Again, if you're not using the Wedge, don't load a directory, just try another disk.

Verify Errors

If you get an error when you VERIFY the LOAD of a program, type NEW and try again. If you still get an error, it means the file did not LOAD properly. (Refer to the section on *LOAD errors*.)

If you get an error when you VERIFY the SAVE of a program, check to make sure the program does not have any visible errors. Check you connections, and try the SAVE again. If it still does not VERIFY refer to the section on SAVE errors.

File Not Found

This means either you typed the name of the file incorrectly, you don't have a file by that name on your disk or, if you are using a dual drive system, you may have requested the wrong file, or if you are using multiple drives, the wrong drive.

ADDITIONAL READINGS

This is a current (10/84) listing of articles from *RUN, Ahoy!, COMPUTE!* and *COMPUTE!'s Gazette* which may be of interest to disk drive users. I have also included a list of books which may be helpful. This list does not constuitute an endorsement of any magazine or book, and is intended only to be a guide, not a definitive listing.

Books

L.R. Carter and E. Huzan, *Teach Yourself Computer Programming with the Commodore 64,* Hodder and Stoughton, Sevenoaks, Kent, England, 1983. American publisher: David McKay Co., Inc., New York, New York.

David Millerer, *Commodore-64 Data Files,* Reston Publishing Co., Inc., Reston, Virginia, 1984.

William B. Sanders, *The Elementary Commodore 64,* Datamost, Inc., Chatsworth, California & Reston Publishing Co., Inc., Reston, Virginia, 1983.

Weber Systems, Inc. Staff, *Commodore 64 User's Handbook,* Ballantine Books, New York, New York, 1983.

Magazine Articles

Ahoy!

Morton Kevelson, "The 1541 Disk Drive: A Guided Tour", No. 2, (February, 1984), pp. 34-38,78.

Michael Kleinert & David Barron, "Random Files on the C-64", No. 3, (March, 1984), pp. 37-38,76.

Michael Kleinert & David Barron, "Programming Relative Files", No. 2, (February, 1984), pp. 27-28,78.

Michael Keinert & David Barron, "Programming Sequential Files", No. 1, (January, 1984), pp. 67-68, 89-90.

COMPUTE!

Robert W. Baker, "Disk Explorer for Commodore", Vol. 5, No. 12, (Dec., 1983), pp. 298-304.

Jim Butterfield, "Advanced Disk Logging on the 64", Vol. 6, No. 4, (April, 1984), pp. 157-158.

Jim Butterfield, "Complex Disk Copies for the 64", Vol. 6, No. 4, (April, 1984), pp. 159-160.

Jim Butterfield, "Commodore DOS Wedges: An Overview;;, Vol. 5, No. 10, (Oct., 1983), pp. 266-270.

Jim Butterfield, "Commodore Files for Beginners, Part 1", Vol. 5, No. 11, (Nov., 1983), pp. 174-182.

Jim Butterfield, "Commodore Files for Beginners, Part 2", Vol. 5, No. 12, (Dec., 1983), pp. 236-240.

Jim Butterfield, "Commodore Files for Beginners, Part 3", Vol. 6, No. 1, (Jan.,1984), pp. 192-194.

Jim Butterfield, "Commodore Files for Beginners, Part 4", Vol. 6, No. 2, (Feb.,1984), pp. 165-167.

Jim Butterfield, "Relative File for the VIC-20 and the Commodore 64, Part 1", Vol. 5, No. 9, (Sept., 1983), pp. 255-256.

COMPUTE!'S GAZETTE

Charles Brannon, "Getting Started with a Disk Drive, Part 1", Vol. 1, No. 5, (November, 1983), pp. 46-52.

Charles Brannon, "Getting Started with a Disk Drive, Part 2", Vol. 1, No. 6, (December, 1983), pp. 60-67.

Charles Brannon, "Getting Started with a Disk Drive, Part 3", Vol. 2, No. 1, (January, 1984), pp. 66-74.

Charles Brannon, "Getting Started with a Disk Drive, Part 4", Vol. 2, No. 2, (February, 1984), pp. 44-52.

Charles Brannon, "Getting Started with a Disk Drive, Part 5", Vol. 2, No. 3, (March, 1984), pp. 106-108, 163-165.

Vern Buis, "VIC/64 Program Lifesaver" (Un-new), Vol. 1, No. 5, (November,1983), pp. 132-134, 203.

Philip Dale, "Disk File Manager", Vol. 1, No. 6, (December,1983), pp. 130-132, 222-224.

Martin Engert, "File Copier", Vol. 2, No. 6, (June, 1984), p. 118.

Larry Isaacs, "64 Explorer:Single-Drive Disk Copying, Part 2", Vol. 1, No. 3, (September, 1983), pp. 96-100, 123-124.
Richard Mansfield, "How to Use Tape and Disk Files", Vol. 1, No. 4, (October, 1983), pp. 118-122.

Wayne Mathews, "Disk Menu", Vol. 1, No. 2, (August, 1983), pp.88, 113-114.

Gerald Sanders, "Disk Purge", Vol. 2, No. 8, (August, 1984), pp.110-111, 133-134.

Bobby Williams, "Power BASIC:Using a 1541 Disk Drive & Commodore 64", Vol. 1, No. 2, (August, 1983), p. 90.

John S. Winn, "Appending Sequential Disk Files", Vol. 2, No. 6, (June, 1984), pp.120-122.

RUN

Robert W. Baker, "Disk Master Revisited", Vol. 1, No. 2, (February, 1984), pp.100-108.

David Brooks, "It's All Relative", Vol. 1, No. 4, (April, 1984), pp.100-107.

David Brooks, "Relatively Speaking", Vol. 1, No. 5, (May, 1984), pp.138-140.

David Brooks, "It's All Relative", Vol. 1, No. 6, (June, 1984), pp.108-117.

Michael Bruossard, "Calling Disk Directories to Order", Vol. 1, No. 4, (April, 1984), pp.122-125.

Thomas Henry, "Disk-O-VIC", Vol. 1, No. 1, (January, 1984), pp.102-118.

Christopher Lampton, "Disk Editor 64", Vol. 1, No. 4, (April, 1984), pp. 71-74.

Cal Overhulser, "Disk-O-64", Vol. 1, No. 6, (June, 1984), pp. 54-56.

John Stillwell, "Database Deluxe", Vol. 1, No. 2, (February, 1984), pp. 48-54.

INDEX

BACKUP 18
BAM 6
Bit Access Map: See BAM
BLOCK 6
Block Allocation Map: See BAM
Block Availability Map: See BAM
Bring Back Files/Disk 7
CHANNEL
  How to Open 15,16
  Reserved 15
COPY
  Disk: See BACKUP
  Disk to Disk 18
  Same Disk 15
  Disk to Tape 18
  Tape to Disk 18
DIRECTORY 6
DISK
  Cleaners 5,6
  Density 5,6
  Handling 8,9
  ID Number 7,11
  Structure 5,6
DISK DRIVE
  Operation 5
  Setup 3
DOS 5,6
ERASE
  File: see SCRATCH
  Disk: see FORMAT
ERROR CHANNEL 16
FILE
  CODES: explanation 9
  General Info 22
  Improper Closing 15,16
  PRG (program) 9,10,12,13
  Random 23
  Random Access 22
  REL (relative) 9,15,23
  SEQ (sequential) 9,15,22
  USR (user) 9,15,22
FORMAT 7,11

HOW TO:
Change the Name of a Program ..... See RENAME
Combine Two Programs: ..... See MERGE
Copy a Program from Tape to disk: ..... See COPY Tape to Disk
Get Printed Copy of Program ..... See OPEN Channel to Printer
Get Rid of a Program: ..... See SCRATCH
Look at a Program: ..... See LIST Program
Make a Copy of your Disk: ..... See BACKUP
Make a Copy of your Program: ..... See COPY
See What's on your Disk: ..... See LOAD Directory
Turn off the Flashing Red Light: ..... See ERROR CHANNEL
Prepare a Blank Disk for Use: ..... See FORMAT
Use a Program on Your Disk: ..... See LOAD Program
See RUN Program
HUB 5
INITIALIZE
Drive 16
Disk: See FORMAT
LIST
Directory 10
Program 9,10,11
LOAD
Directory 9,11
Program:
Ready-made Disk 10
Your Own Disk 12
MERGE 15
NEW (Disk): See FORMAT
OPEN Channel to Printer 17
PATTERN MATCHING 17,18
RENAME 15
RUN
Program:
Ready-made Disk 11
Your Own Disk 12,13
SAVE 12,13
SAVE@0 13
SAVE and Replace: See SAVE@0
SCRATCH 7,14
SECTOR: See BLOCK
TRACK 5,6
VALIDATE 15
VERIFY 10,12
WEDGE
Commands 19,20
Loading 19,20
Running 19
Why Use It 19
WILD-CARD MATCHING: see PATTERN MATCHING

AT-A-GLANCE REFERENCE

COPY A FILE/PROGRAM:

```
type OPEN 15,8,15                                    [RETURN]
     PRINT#15, "CO:newfile = 0:oldfile"              [RETURN]
     CLOSE 15                                        [RETURN]
or
     OPEN 15,8,15:PRINT#15, "CO:newfile = 0:oldfile":
     CLOSE 15 [RETURN]
```

COPY TO COMBINE FILES:

```
(combine 2 to 4 files)
type OPEN 15,8,15                                    [RETURN]
     PRINT#15, "CO:newfile = 0:oldfile1,0:oldfile2,0:
     oldflie3,0:oldfile4"                            [RETURN]
     CLOSE 15                                        [RETURN]
```

RENAME A FILE:

```
(will not work on an open file)
type OPEN 15,8,15                                    [RETURN]
     PRINT#15, "R0:newname = 0:oldname"              [RETURN]
     CLOSE 15                                        [RETURN]
or
     OPEN 15,8,15:PRINT#15, "R0:newname = 0:oldname":
     CLOSE 15 [RETURN]
```

Remember, open the error channel (15) first and close it last. It automatically closes your other sequential files when it closes.