NorthStar

V. 1.300

TurboDOS®
User's Guide

To reorder the TurboDOS User's Guide, order part number 03241.
To reorder the TurboDOS Reference Manual,
order part number 03841.

Copyright Notice

Trademark Notice

Disclaimer

Purpose

We've designed this User's Guide to make it easy for you to learn how to use the TurboDOS operating system. This document tells you what TurboDOS does, how to use its various facilities, and what to do in case of errors. It also describes each TurboDOS command in detail. The information presented applies to both Z80 and 8086 TurboDOS.

Assumptions

In writing this guide, we've assumed that you have a pre-configured copy of TurboDOS all ready to run on your computer. Also, you need to know some basic things about your computer equipment: how to hook it up and start it running, and how to handle the disks it uses. We haven't assumed you are a programmer or an experienced computer user, however.

Organization

This guide starts with a section that explains some of the basics: what TurboDOS is, what it does, how to get it started, enter commands, and correct errors.

The next three sections explain the principal concepts of TurboDOS in more detail. There are sections on files and disks, printing, and processing.

There is a reference section that describes each TurboDOS command in detail. This section is organized alphabetically for easy reference.

Finally, there is a glossary and an index.

SOME BASICS

In this section you will learn what TurboDOS does, what kind of hardware it runs on, how to start it up, and how to enter commands and correct errors.

Let's start with the obvious question...

What is TurboDOS?

TurboDOS is a collection of programs designed to make your computer run more efficiently, and to make it easier for you to run other programs, create and manage files, and use the peripheral devices (such as the console, printer, and disk drives) attached to your computer.

TurboDOS is an "operating system", the technical term for a program which supervises the operation of other programs. TurboDOS itself is not specialized for any particular application (such as word processing, accounting, or statistics). Instead, TurboDOS works together with whatever specialized application program you care to run, and takes care of whatever file and device management functions your particular application program requests.

Now let's examine what makes TurboDOS different from other operating systems.

Why Use TurboDOS?

Compared to other microcomputer operating systems, TurboDOS is distinguished by:

- Networking Capability
- CP/M Compatibility
- Superior Performance
- Ability to Handle Large Files and Disks
- Enhanced Reliability
- Print Spooling

We'll begin with networking.

Networking

Compared with other operating systems, TurboDOS perhaps is most distinguished by its ability to coordinate a "network" of interconnected microcomputers. Generally, a separate microcomputer supports each user's video console. One or more additional microcomputers manage the disks, printers, and other shared devices of the system.

Since this approach provides a microcomputer dedicated to each user, TurboDOS is able to support a large number of simultaneous users with excellent performance and minimal interaction. You will discover this networking approach offers far better performance and reliability than does any system that relies on "time-sharing" one processor among many users.

CP/M Compatibility

TurboDOS lets you run any program designed to operate with Digital Research's popular CP/M and MP/M operating systems. This means that you can choose from a vast library of available software packages: financial applications, word processors, spelling checkers, spreadsheets, language processors, and programming tools.

TurboDOS runs virtually any CP/M program without modification, and accepts standard CP/M-format disks. TurboDOS is fully compatible with CP/M-80 2.2 and CP/M-86 1.1. In addition, TurboDOS provides compatibility with CP/M Plus 3.1, MP/M II 2.1 and MP/M-86 2.1 in file and record locks, system date and time, and several other selected areas.

Performance

Independent benchmark comparisons have shown that file-intensive applications perform much better with TurboDOS than with CP/M or MP/M. You'll benefit from this performance improvement with virtually any business application. File processing averages at least three times as fast, sometimes much more. The more users on your system, the greater the performance advantage becomes.

Disk Capacity

Business applications are often limited by the available disk capacity of your computer.

TurboDOS is designed to handle big hard disks and big files. CP/M has internal limitations which prevent it from supporting disks or files larger than 8 megabytes. Disk drives larger than 8 megabytes have to be partitioned into segments, making them awkward to use. In contrast, TurboDOS supports disk drives to 1,000+ megabytes without partitioning, and supports files up to 134 megabytes.

Reliability

Integrity of data and graceful recovery from errors are critical to any serious data processing activity. TurboDOS offers vital improvements in these important areas.

TurboDOS does read-after-write of all disk updates. In other words, whenever it writes information out to disk, it immediately reads it back to make sure it got recorded okay. (Most microcomputers don't do this!)

Reliability (Continued)

Whenever errors are detected, TurboDOS gives you meaningful diagnostic messages, and lets you chose from a variety of recovery options. In the event of a disk error, for example, you can choose one of these alternatives:

- Retry the disk operation again
- Ignore the error and continue processing
- Abort the program

Print Spooling

The slowest component of your computer system is undoubtedly the printer. If your applications involve much printing, you may find that the printer is the bottleneck that limits how much work you can get done on your system. To help solve this problem, TurboDOS includes an automatic "print spooling" facility that lets you get on with the next processing task without having to wait for printing from the previous task to finish.

Here's how it works. When your application program tries to send characters to the printer, TurboDOS intercepts them and saves them in a disk file instead. (This is called "spooling".) When the program is done, TurboDOS automatically starts printing the contents of the file. (This is called "despooling".) At the same time, it lets you go ahead and run your next program.

In a multi-user system, several users may try to print at the same time. In this case, TurboDOS automatically queues the print files on a first-come first-served basis.

Getting Started

To get TurboDOS started, it is necessary to read a copy of the operating system from your TurboDOS system disk into your computer's memory. This process is variously known as "cold start".

Cold Start

The exact cold-start procedure depends on the kind of hardware you are using.

Usually, all you need to do is turn on the computer's power switch, or press the computer's "reset" button (if the power is already on). If your TurboDOS system disk is a floppy disk or other type of removable media, you need to load it into the disk drive. At this point, TurboDOS will load into memory automatically, and sign on.

Sign On

On a single-user system (or a multi-user system with no log-on required), you will see the following display on your console as soon as TurboDOS is loaded:

```
TurboDOS 1.3x, Copyright 1983 etc.
0A}
```

The "sign-on" message tells you which version of TurboDOS you are using. The "0A}" is the "command prompt" that tells you TurboDOS is ready to accept a command from your keyboard. It also tells you that user number zero and drive A are currently selected (more about this later).

Multi-user systems usually require you to identify yourself before you may proceed. Let's see how this is done.

Log On

On most multi-user systems, you must "log on" by entering a valid user identification, password, and possibly other information:

```
TurboDOS 1.3x, Copyright 1983 etc.
System log on
Enter user id: BARBARA
Enter password: SHAZAM
Enter date: 15 APR 84
Enter time: 14:25:30
Enter activity: PAYROLL
5A}
```

In the example above (and throughout the remainder of this document), underlines are used to indicate the items you need to key in. The rest is displayed by TurboDOS. After you have typed each response, you have to press the RETURN (or ENTER) key on your keyboard before TurboDOS will process the item.

The way TurboDOS validates user id's and passwords is described later on (see the details of the LOGON command). The date and time are not requested if the system clock was previously set. The activity description is requested only if a system log file has been set up.

As soon as you get through the log-on procedure successfully, TurboDOS displays its "command prompt" to let you know it's ready to accept your first command.

Next, we'll learn how to enter commands.

Entering Commands

TurboDOS does only what you tell it to do. The way you tell it what to do is by entering commands on your console keyboard.

Whenever you enter a command, you are actually asking TurboDOS to load a particular program from disk into your computer's memory and run it. TurboDOS comes with more than 30 standard commands (programs) that are described in detail in a later section of this guide. You can also enter commands to run application programs, or programs you have written yourself.

Command Format

A TurboDOS command always contains a command name which specifies the program to be run. The simplest commands consist only of the command name, followed by the RETURN (or ENTER) key. For example:

```
0A}DIR
```

is a commonly-used command which displays a directory of the files on the currently-selected user number (zero) and disk drive (drive A). Remember that the command name "DIR" is actually the name of a program on disk. Don't forget the RETURN key at the end of each command.

Frequently, the command name is followed by additional command information (drive letter, file name, option, etc.) to be passed to the command program. For example:

```
0A}DIR B:
```

displays the directory for drive B.

Command Format (Continued)

Another example:

```
0A}COPY A:ANYFILE B:
```

is a command to copy a file named "ANYFILE" from drive A to drive B. "COPY" is the name of another command program.

Multiple Commands

You may enter several TurboDOS commands at one time. To do this, separate the commands with the character \ (backslant), and press RETURN after the last command. For example:

```
0A}DIR B:\DIR C:\DIR D:
```

is a string of three commands to display the directories of drives B, C and D in sequence.

You can string as many commands together as you like, limited only by the size of the TurboDOS command buffer. Normally, the command buffer is big enough for you to type a command string two lines long.

When you enter a string of commands, TurboDOS executes each command in sequence, one after another. Each command but the first is displayed on your console as it is executed.

Correcting Errors

As you type characters on the keyboard, they appear on the screen of your console. An indicator called the "cursor" (usually a box or underline) shows you where the next character will appear.

If you make a typing mistake and you notice it before you press RETURN, you can correct it easily. Use the BACKSPACE or DEL key to delete the incorrect character(s) and move the cursor left. Then continue typing.

If your keyboard doesn't have a BACKSPACE or DEL key, you can accomplish the same thing by typing CTRL-H. To type CTRL-H, you hold down the CTRL key while typing the letter H.

If you want to delete the entire line you've typed, you could keep typing BACKSPACE until it's all gone. An easier way is to type CTRL-X or CTRL-U (hold down CTRL while typing X or U), which accomplishes the same thing.

If you accidentally misspell the name of a command, and hit RETURN before realizing your mistake, TurboDOS will tell you that it can't find a command with that name:

```
0A}DIT B:
DIT B: <-- Command not found
0A}
```

and you can then retype the intended command. Likewise, if you enter a command TurboDOS can't figure out, it will let you know:

```
0A}DIRB:
DIRB: <-- Invalid command
0A}
```

Attention Requests

You can suspend the execution of a program or command at any time by typing the "attention" character on your console keyboard, the BREAK key. TurboDOS will "beep" to acknowledge that it has received your attention request and suspended execution of your program.

After an attention request, your program will remain suspended until you type one of the following attention responses:

CTRL-Q (resume) simply restarts execution of your program at the point it was interrupted. You will find this attention/resume sequence useful to prevent displayed information from scrolling off the top of your console screen before you have time to read it.

CTRL-C (abort) cancels program execution, and returns you to the TurboDOS command prompt.

CTRL-P (echo-print) restarts program execution and causes all subsequent console output also to be echoed on the printer. A second attention/echo sequence turns off echoing of console output to the printer. This feature is useful for getting hardcopy of your console interactions with the system.

CTRL-L (end-print) restarts program execution after signalling the end of the current print job. This is useful primarily when spooling, because it causes all accumulated print output to be immediately queued for printing.

Summary

This section has covered some of the basics you need to know about TurboDOS. You've learned what an operating system is, how TurboDOS compares with other microcomputer operating systems, and what kind of hardware it runs on. You've also learned how to load TurboDOS from a cold start, log on, enter commands and command strings, correct typing errors, and make attention requests.

Our next topic is one of the most important in this guide: files and disks.

FILES AND DISKS

In this section, you will learn more about how TurboDOS does its most important job: managing your files and disks.

What is a File?

A "file" is a collection of related information stored on disk. Files may contain data (numbers, text, formulas, pictures, and so on) or computer programs.

You can think of a file on disk as being like a file folder in a file cabinet. When you store information in a file cabinet, you use file folders to group related information together. For instance, one folder might contain information about company employees (names, addresses, telephone numbers, and so forth). You might label this folder "Employee Data" so you can find it later.

Using your computer's disk instead of a file cabinet, you could set up a disk file containing the employee information. You might give this file the name EMPLOYEE.DAT. To keep track of your files, you have to give each one a name.

Identifying Files

TurboDOS keeps track of your files by name. It maintains a directory of the files on each disk. Whenever you want to access a particular file, you simply identify the one you want by name.

Names and Types

Within limits, you can give your files any names you want. Each file name can be up to eight characters long. Try to make up names that will remind you what each file contains:

```
EMPLOYEE    CONTRACT    LOANCALC
CUSTOMER    TAXINFO     STARTREK
PAYROLL     USRGUIDE    PAYCHECK
```

As you create more and more files, you will find they begin to fall into natural categories (commands, BASIC programs, documents, data base files, and so forth). To help you group similar files together, you may add an optional extension (called a "file type") to your file names. The file type can be up to three characters long, and is always separated from the file name with a period:

```
EMPLOYEE.DAT   CONTRACT.DOC   LOANCALC.BAS
CUSTOMER.DAT   TAXINFO.DOC    STARTREK.BAS
PAYROLL.DAT    USRGUIDE.DOC   PAYCHECK.BAS
```

Certain file types have special meaning to TurboDOS. For example, Z80 commands (programs) are kept in .COM files, and 8086 commands are kept in .CMD files.

You can make up file names and file types using any desired combination of letters A-Z and digits 0-9. It's a good idea to stay away from punctuation marks.

Directories

To keep track of your files, TurboDOS maintains a file directory on each disk. For each file, the directory contains the name, type and attributes of the file, what user it belongs to, how big it is, and where it is located on the disk.

You can use the DIR command to display or print the file directory on any of your disks. For example:

```
0A}DIR

THISDISK.LAB  15-Apr-84  900K REMAINING
22 FILES      0A:*.*     324K DISPLAYED

CONTRACT.DOC 12K        STARTREK.BAS  8K
CUSTOMER.DAT 58K        TAXINFO .DOC  2K
EMPLOYEE.DAT 28K        USRGUIDE.DOC 96K
etc.
0A}
```

The DIR command displays the files in alphabetical order by name, and shows you the size of each one in kilobytes. (A kilobyte is 1024 bytes.) It also shows you how much unused space remains on the disk, and other pertinent information.

Wild-Cards

Sometimes it's convenient to be able to refer to an entire group of files, such as "all payroll data files" or "all BASIC program files". For this purpose, TurboDOS provides two special characters ? and * that can be used in file names and types.

You can use the ? in a file name or type as a "wild-card" to match any character in the corresponding position. You can use the * to indicate that all remaining character positions of the file name or type are wild-cards.

For example, the command:

```
0A}DIR PAY?FILE.DAT
```

might list the following files:

```
PAY0FILE.DAT     PAYAFILE.DAT
PAY5FILE.DAT     PAYXFILE.DAT
PAY9FILE.DAT     PAYZFILE.DAT
```

while the command:

```
0A}DELETE *.BAS
```

would delete all files of type .BAS from the disk.

You'll find these wild-cards especially useful in connection with certain TurboDOS commands (such as COPY, DIR, DELETE, and RENAME) for processing a group of files without having to name each one individually.

Drive Selection

Your computer system may be equipped with several disk drives (up to 16 of them). Each drive is assigned a letter A-P. In its command prompt, TurboDOS reminds you which drive it is currently working on.

When you refer to a file by name, TurboDOS normally looks for that file on the current drive. To access a file on a different drive, you have to enter a drive letter and a colon in front of the file name.

For example, if you enter:

```
0A}DELETE B:DOCUMENT.TXT
```

TurboDOS will look for the file DOCUMENT.TXT on drive B, even though the current drive is drive A.

You can also use a drive specification in front of a command keyword to indicate that you want TurboDOS to load the command from a particular drive:

```
0A}C:DELETE B:STARTREK.BAS
```

In this example, TurboDOS would look for DELETE.COM (or DELETE.CMD) on drive C, and for STARTREK.BAS on drive B.

Drive Selection (Continued)

If you need to access many commands or files on the same drive, you might find it convenient to change the current drive so that you don't have to keep specifying the drive explicitly on every reference. To change the current drive, simply type the drive letter, a colon, and then RETURN:

```
0A}B:
0B}
```

Notice that TurboDOS changes its command prompt to indicate the new current drive.

Working with Files

At this point, let's see how you can accomplish some basic file operations using TurboDOS. These include creating, copying, renaming and deleting files.

Creating Files

TurboDOS does not have any special command to create a file. Most of the time, you create new files by running application programs. For example, to create a text file you would probably use a text editor or word processing program. You can also create files by making copies of other files.

When a program creates a new file, the file starts out empty. As the program writes information to the file, TurboDOS automatically allocates additional disk space. A file can grow as large as needed, limited only by the size of the disk. When a file is deleted, the disk space previously occupied by the file becomes available again.

Copying Files

To copy a file, use the COPY command:

```
0A}COPY ORIGINAL.BAS DUPLICAT.BAS
0A:ORIGINAL.BAS copied to 0A:DUPLICAT.BAS
0A}
```

If the second (destination) file name is not given in the COPY command, then the first (source) name is used:

```
0A}COPY ORIGINAL.BAS C:
0A:ORIGINAL.BAS copied to 0C:ORIGINAL.BAS
0A}
```

You can use wild-cards to copy a group of files with a single command:

```
0A}COPY *.BAS C:
Confirm individual files (y/n)? N
0A:LOANCALC.BAS copied to 0C:LOANCALC.BAS
0A:ORIGINAL.BAS copied to 0C:ORIGINAL.BAS
0A:STARTREK.BAS copied to 0C:STARTREK.BAS
0A}
```

The COPY command is one of the most useful in TurboDOS. It has numerous features and options that you'll want to study carefully when they're described later in this guide.

Renaming Files

To change the name of a file, use the RENAME command:

```
0A}RENAME LOANCALC.BAS LC.BAS
0A:LOANCALC.BAS renamed 0A:LC      .BAS
0A}
```

Using wild-cards, you can rename a group of files:

```
0A}RENAME STARTREK.* TREK.*
Confirm individual files (y/n)? N
0A:STARTREK.BAS renamed 0A:TREK    .BAS
0A:STARTREK.COM renamed 0A:TREK    .COM
0A:STARTREK.DOC renamed 0A:TREK    .DOC
0A:STARTREK.REL renamed 0A:TREK    .REL
0A}
```

Deleting Files

To erase a file from disk completely, use the DELETE command:

```
0A}DELETE ORIGINAL.BAS
0A:ORIGINAL.BAS deleted
0A}
```

Like COPY and RENAME, the DELETE command can be used with wild-cards to delete a group of files with one command. You should be especially careful when using DELETE, for obvious reasons.

Backing Up

It is extremely important for you to make backup copies of all your disks. Disks can be damaged and files can be erased as a result of human error or equipment failure. Such an event can be catastrophic if you have no backup copy of the lost disks or files. However, if you have been conscientious about making backup copies, you can recover your programs and data easily.

Always make a working copy of any new software package you purchase, and save the original. If the working copy is destroyed for any reason, you can always restore it from the original.

Always make frequent backup copies of any programs and data files you are working on. As a general rule, you should make daily backup copies of all files you created or changed during the day. In many instances, it is prudent to make copies more frequently than once a day.

There are two ways to make backup copies: using the BACKUP command or the COPY command.

Using BACKUP

The BACKUP command creates an exact duplicate of the contents of one disk on another disk:

```
0A}BACKUP A: B:
Insert source disk in drive A
Insert destination disk in drive B
Enter <cr> to begin copying: [RETURN]
.....................................
Successful copy
0A}
```

Using Backup (Continued)

The BACKUP command can be used only to copy an entire disk to another disk of exactly the same kind and format. BACKUP is the fastest method of copying a disk, because it copies everything in one operation.

Using COPY

The COPY command can also be used for making backup copies. You can use COPY to copy all files from one disk to another:

```
0A}COPY A: B: ;N
0A:ALPHA    .COM copied to 0B:ALPHA    .COM
0A:BETA     .BAS copied to 0B:BETA     .BAS
                    :
                    :
0A:ZULU     .DAT copied to 0B:ZULU     .DAT
0A}
```

COPY is slower than BACKUP because it copies one file at a time. COPY has some advantages, however. You can copy from one kind of disk to another -- from a hard disk to a floppy disk, for example. You can copy selected files or groups of files if you wish. In addition, the COPY command has an "archived" option for copying only those files that have been changed since last time you backed up.

Working with Disks

So far, we've seen how TurboDOS works with files stored on disk. A single disk can hold dozens of files, sometimes hundreds of them. However, certain TurboDOS commands and facilities deal with the disk itself, rather than with the individual files stored on it.

Kinds of Disks

A disk is a rotating magnetic medium used by a computer system to store programs and data. Information is recorded on disk magnetically, much as voice or music is recorded on audio tape. The term "disk" applies to several kinds of storage media: floppy disks, removable cartridge disks, and fixed hard disks.

Floppy disks (also called "diskettes") are inexpensive, and often the only kind of storage available on small microcomputer systems. They are removable disks which use a flexible magnetic medium that spins inside a cardboard jacket. Floppy disks come in 8-inch and 5.25-inch diameter sizes (even smaller ones are being developed). They may be recorded on one or both sides, and commonly have capacities of 150K to 1,500K (kilobytes).

Cartridge disks are also removable, but they use a rigid ("hard") medium that allows much more data to be stored on each disk. They are considerably more expensive than floppies, so their use on microcomputer systems has been somewhat limited. Cartridge disks come in 14-inch, 8-inch, and 5.25-inch diameters, and typically hold 2 to 20 megabytes (a megabyte is 1024K, more than a million bytes).

Kinds of Disks (Continued)

Fixed hard disks (sometimes called "Winchester" disks) use one or several rigid magnetic platters sealed inside an airtight container. They provide a reliable and economical means of storing a lot of information (typically 5 to 50 megabytes), and are rapidly becoming popular in medium-to-large microcomputer systems. The disadvantage of such disks is that they are non-removable. Consequently, some other kind of removable medium (floppy disk, cartridge disk, or tape) is almost always required for backup and archive purposes.

Disk Formats

Information is recorded on disks along concentric circular "tracks", somewhat like the grooves on a phonograph record. Each track is further subdivided into fixed-length "sectors". The "format" of a disk refers to how many tracks are recorded on the disk (on either one or both sides), how many sectors are recorded on each track, and how many bytes are recorded in each sector.

For some kinds of disks ("hard-sectored"), the format is fixed by the hardware itself. For others ("soft-sectored"), the format must be pre-recorded onto the disk by means of the TurboDOS FORMAT command before the disk can be used. Most floppy disks and many hard disks are soft-sectored.

For maximum capacity and performance, floppy disks used with TurboDOS are generally formatted with large sector sizes (typically 512 bytes). However, TurboDOS can also accomodate standard CP/M formats. TurboDOS determines automatically whether a floppy disk is recorded in CP/M or TurboDOS format. TurboDOS-format diskettes run very much faster then CP/M diskettes. The FORMAT command can initialize a disk in either CP/M or TurboDOS format.

Disk Formats
(Continued)

You must use FORMAT to format every floppy disk before it can be used by TurboDOS for the first time:

```
0A}FORMAT B:
Enter Format Type:
1 = North Star CP/M Double-Density
          (Single-Sided)
2 = North Star CP/M Quad-Capacity
          (Double-Sided)
3 = North Star TurboDOS Double-Density
          (Single-Sided)
4 = North Star TurboDOS Quad-Capacity
          (Double-Sided)
  = 2
Insert disk to be formatted in drive B
Enter <cr> to begin formatting: [RETURN]
Starting format pass:
.....
Starting verify pass:
.....
0A}
```

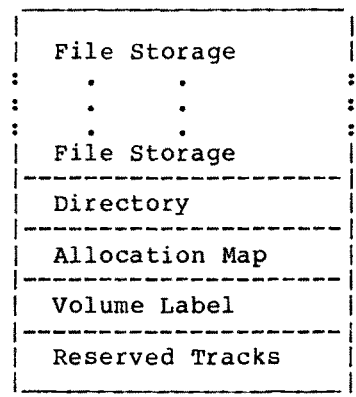
It is also a good idea to use the ERASEDIR command to create an empty directory on the newly-formatted disk, and to use the LABEL command to give a name to the disk:

```
0A}ERASEDIR B:
Hashed directory desired (Y/N)? N
OK to erase directory on drive B? Y
Directory erased, non-hashed
0A}LABEL B:PAYROLL.DAT
Disk label written
0A}
```

You must use this procedure whenever a new disk is placed into service for the first time. You may also use this procedure whenever you want to erase everything from a used disk and start out fresh. Be careful -- once you use FORMAT or ERASEDIR, any files previously stored on the disk are lost forever.

Disk Organization

You don't really need to know how disks are organized in order to use TurboDOS, but it can be helpful in understanding certain disk-oriented commands. Each disk is organized by TurboDOS into five areas illustrated below:



The "reserved tracks" are required by certain hardware configurations to support system start-up, but not otherwise used by TurboDOS. If your equipment requires reserved tracks, you can use the BOOT command to write information on them.

The "volume label" contains the name you have assigned to the disk. The LABEL command lets you give a name to each of your disks.

The "allocation map" is the area in which TurboDOS keeps track of which portions of the disk storage space are occupied with information, and which portions are free. You can use the FIXMAP command to reconstitute this critical information whenever you suspect it might have become corrupted. It is a good idea to use FIXMAP after any "crash" or other unexplained system malfunction.

Disk Organization (Continued)

The "directory" is a table of contents which lists all files stored on the disk. For each file, there is a directory entry containing the file's name, type, attributes, ownership, size, and location on the disk. The DIR command lets you display or print the directory. The FIXDIR command lets you reorganize the directory when frequent additions and deletions of files cause directory access to become less efficient.

All the rest of the disk (most of it) is available to hold your files. You can obtain details about the format and organization of any disk by using the DRIVE command:

```
0A}DRIVE C:
Disk drive characteristics, drive C
Maximum data capacity      : 1224K
Allocation block size      : 2048
Number of directory entries:  256
Physical sector size       : 1024
Physical sectors per track :   16
Physical tracks per disk   :   77
Number of reserved tracks  :    0
Media is removable
0A}
```

Unlike TurboDOS, CP/M does not maintain a volume label or allocation map on the disks it creates. When a CP/M disk is first accessed by TurboDOS, the first few CP/M directory entries are automatically relocated to the end of the directory in order to make room for the label and map. When a TurboDOS disk is accessed by CP/M, the label and map appear to be ordinary deleted directory entries. The result is that disks can be moved freely between CP/M and TurboDOS without concern about the differences in organization.

Directory Formats

In addition to CP/M-compatible "linear" directories, TurboDOS also supports an optional "hashed" directory format that makes look-up in large directories much faster.

The standard linear directory format is compatible with CP/M, and is searched sequentially. Because of this, look-up speed gets slower as the directory gets bigger. For hard disks with big directories, look-up time can get painfully slow.

The optional hashed directory format uses a "hashing algorithm", a sophisticated technique for making look-up on large directories much faster. A hashed directory may be used on any disk, but is especially suited for use on hard disks with big directories. Hashed directories are not compatible with CP/M.

To initialize a disk to use this optional hashed directory format, use the ERASEDIR command. Alternatively, to change an existing directory from linear to hashed (or vice-versa), use the FIXDIR command:

```
0A}FIXDIR C:
Hashed directory desired (Y/N)? Y
OK to reorganize directory on drive C? Y
Directory reorganized, hashed
0A}
```

When you display a hashed directory using DIR, the symbol "(H)" appears in the heading.

Whether the directory is linear or hashed, searches involving "wild cards" have to be done linearly. In fact, such wild-card searches are typically slower if the directory is hashed.

Changing Disks

Multi-user systems require special care when changing floppy disks (or other removable media). First, you should check to make sure that no other user is using the disk you want to change. Then use the CHANGE command to let TurboDOS know you want to change disks:

```
0A}CHANGE
Enter drive(s) to be changed: BCD
Change drive(s) BCD
Enter <CR> to continue
```

Wait until the CHANGE command tells you it's okay to change the disks (as above). Then remove and replace disks as required. Finally, press RETURN to advise you are done with the change. Remember, never remove a disk without first entering a CHANGE command.

Disk Errors

If you hear a "beep" and see a diagnostic message on your console such as:

```
Read Error, Drive A, Track 0, Sector 2
[Retry, Ignore, Abort]

Write Error, Drive B, Track 5, Sector 16
[Retry, Ignore, Abort]
```

it means that TurboDOS could not read or write the specified disk sector even after a number of retries. When you see such a message, you must choose one of three recovery options (Retry, Ignore, Abort) by keying the letter R, I, or A.

Disk Errors
(Continued)

If you key R (retry), TurboDOS will try the read or write operation several more times. If it is still unsuccessful, you will get another message.

If you key I (ignore), processing will continue as if the error had not occurred. This option is not recommended in most situations because it causes processing of invalid data and may lead to other errors.

If you key A (abort), TurboDOS terminates the program you were running when the error occurred.

If you hear a "beep" and see this message on your console:

```
Not Ready Error, Drive C [Retry, Abort]
```

it means TurboDOS could not access the selected drive for one of the following reasons:

- there's no such drive on your system
- the drive isn't ready to operate
- no disk is mounted in the drive
- the disk hasn't been formatted
- the disk format is unrecognizable
- you forgot to type SERVER before FORMAT, VERIFY, etc.

Again, you must select the desired recovery option by keying R or A.

User Numbers

You know already that TurboDOS maintains a directory on each disk which lists all files on the disk. Within the directory, TurboDOS provides up to 32 separate file libraries corresponding to user numbers 0-31. These libraries are especially useful in a multi-user system, since they allow you to maintain separation between files belonging to different users or applications.

When you first log on to a multi-user TurboDOS system, the system assigns you to one of these user numbers according to the user id that you specify. You can see which user number you have been assigned by looking at the command prompt. For example:

```
5B}
```

tells you that your user number is five. Generally, user 0 is reserved for global files (explained shortly) and user 31 is reserved for log-on security, leaving 1-30 for general use.

Ordinarily, you are confined to your assigned user number from log on until log off. Most file operations (creating, renaming, deleting, searching, etc.) are confined to the library which corresponds to your user number. For example, if you enter the command:

```
5B}DIR
```

you will get a directory of only those files in the user 5 library.

User Numbers (Continued)

The log-on mechanism of TurboDOS permits certain user id's to be "privileged", allowing those users to access various protected facilities of TurboDOS. If you qualify as a "privileged user", you can change user numbers anytime you like with the USER command:

```
5B}USER 12
12B}
```

In most systems, you are automatically logged on to user zero as a privileged user. You can then change to any other user number with the USER command.

File Attributes

File attributes may be used to control how a file can be accessed. TurboDOS defines four such attributes: Read-only, Global, Archived, and FIFO. Each of these attributes is explained in detail below.

You can set and clear file attributes with the SET command. For example:

```
0A}SET *.COM ;+RG-A
```

sets all .COM files to Read-only, Global, and not Archived. The SHOW command lets you see the settings of file attributes.

Read-Only File

TurboDOS lets you protect important files and disks against accidental destruction by marking them with the "read-only" attribute. A read-only file cannot be written, deleted or renamed.

To delete, rename or modify a read-only file intentionally, you must first clear the read-only attribute.

Global File

If you save a file under user 0 and mark it with the "global" attribute, you can access that file from any user number. This provides a convenient way to make selected files available to all users. For example, you'd probably want to do this with most TurboDOS commands and other common programs. (Otherwise, you'd have to put duplicate copies under each user number, wasting disk space.)

Remember, global files must be in the user 0 library -- otherwise, the global attribute has no effect.

Archived File

The "archived" attribute works together with the COPY command to provide a convenient mechanism for doing incremental file backup. This attribute is set automatically whenever a file is archived by the COPY command, and cleared automatically whenever a file is written or renamed. When the COPY command is used for archiving, it automatically skips over files which have the archived attribute set. As a result, only files that have been written or renamed since the last archiving cycle are copied.

This incremental backup facility is particularly useful for backing up large fixed hard disks, where copying all files would be prohibitively slow.

FIFO File

The "FIFO" attribute causes a file to be accessed by TurboDOS using a special first-in-first-out access method. FIFO files are especially convenient for inter-process and inter-user communications, and are described in more detail in the Programmer's Guide.

Read-Only Drive

TurboDOS allows drives to be marked read-only much in the same fashion as files. Another variation of the SET command is used:

```
0A}SET C: ;+R
Drive C set to read-only
0A}SET C: ;-R
Drive C set to read/write
0A}
```

Setting a drive read-only prevents TurboDOS from making any attempt to write to that drive. This is a good way to protect important disks against accidental destruction while making backup copies, for instance.

File Searches

Whenever you ask to access a file, TurboDOS first searches the library corresponding to the current user number. If the file is not found, TurboDOS then searches the user 0 library to see if the file is there and is marked as global. For example, if you entered this command:

```
5B}TYPE C:TAXINFO.DOC
```

TurboDOS would first search the user 5 library on drive C. If it can't find TAXINFO.DOC there, then it searches the user 0 directory on drive C, looking for TAXINFO.DOC marked with the global attribute. If both searches are unsuccessful, the TYPE command displays the message "File Not Found".

The search procedure for command files (programs) is similar, but with one added complication. TurboDOS can be configured with a specified "search drive" which is searched automatically if you don't specify an explicit drive letter, and if the normal search of the current drive fails. Using the previous example, if your system is configured with drive A as the "search drive", TurboDOS would search for the command file TYPE.COM (or TYPE.CMD) in the following sequence:

- drive B, user 5
- drive B, user 0 (global)
- drive A, user 5
- drive A, user 0 (global)

before giving up and displaying the message "Command Not Found".

File Sharing

In a multi-user TurboDOS system, it is quite possible that two or more users may want to access the same file at the same time. This can happen if the users are logged on to the same user number, or if they are accessing the same global file. TurboDOS has various facilities to regulate such file sharing.

File Locks

As a general rule, TurboDOS allows any number of users to read a file at the same time, but only one user may write the file. (This is called the "permissive" rule.) If a second user tries to write the file before the first user is finished, the second user will receive an error message. The exact message depends upon the application programs being run, but is usually something like "cannot write file" or "disk full".

Alternatively, it is possible to set up TurboDOS so that only one user can access any file at a time. (This is called the "exclusive" rule.) If your system is set up in this fashion, and if a second user tries to access the file before the first is finished, the second user will receive an error message, usually something like "cannot open file" or "file not found".

Record Locks

TurboDOS also has record locking facilities that make it possible for several users to access and update common files at the same time. Record locking is not automatic. Programs must make explicit requests for TurboDOS to open files in a special "shared" mode, and to lock and unlock records.

Remember, simultaneous updating requires application programs specially-written for a multi-user environment. Ordinary single-user applications will not do.

Summary

This section has explained how TurboDOS manages your files on disk. You've learned what files are, how they are identified by name, and how you can create, copy, rename and delete them. You know how TurboDOS disks are organized, and the proper procedures for formatting disks, changing disks, and recovering from disk errors. You understand the importance of making frequent backup copies of your important programs and data files. And you know about user numbers, file attributes, and file sharing. Congratulations!

Next, we look at how TurboDOS handles printing and print spooling.

PRINTING

This section teaches you all about printing under TurboDOS, including how to use its print spooling capabilities to best advantage. The printing facilies of TurboDOS are rather elaborate, so there's quite a bit to learn.

Printing Methods

We'll start by describing the various printing methods supported by TurboDOS. Later, we'll go into detail about the commands used to control printing.

Your system may be equipped with several printers (up to 16 of them), all of which may be in-use simultaneously. Each printer is assigned a letter A-P (just as in the case of disk drives). You can control which printer you want to use at any given time. In a multi-user system, each user can control his print routing independently.

There are two fundamentally different methods of printing in TurboDOS: direct and spooled.

Direct Printing

Print output may be routed directly to any specified printer on a character-by-character basis. This is the simplest method of printing in TurboDOS, and the only method supported by CP/M. Direct printing is useful for very long print jobs (e.g., overnight), and for certain special situations such as single-sheet printing.

Direct printing has some drawbacks, however. For one thing, when you use direct printing, your console is tied up until the print job has finished. Printers are generally the slowest components of any computer system, and you will find they are often a bottleneck limiting how much work you can get done.

Direct Printing (Continued)

Another drawback of direct printing is that it is very awkward to use in a multi-user environment. If two users attempt to print directly to the same printer at the same time, the result is a merged printout that is not likely to be of much use to either user. Thus, direct printing requires that users carefully coordinate among themselves to avoid such conflicts.

You can eliminate both of these problems by using spooled printing.

Spooled Printing

When you select spooled printing, TurboDOS intercepts the print output from your program and saves it in a print file on disk. This process is called "spooling", and the resulting print file is sometimes called a "spool file".

When the print job is done, TurboDOS automatically starts printing the contents of the print file. This process is called "de-spooling", and takes place in the background independent of your console. You can go ahead and run your next program without waiting for the printing to finish. When the printing is finished, the print file is deleted automatically.

In a multi-user environment, several users may be generating spooled print output at the same time without any interference. As their jobs finish, TurboDOS automatically queues the print files for de-spooling on a first-come first-served basis.

Print Jobs

When a program generates its first character of print output, a new "print job" begins. If spooled printing is in effect, TurboDOS creates a new print file automatically at this point. Subsequent print output is spooled to this print file until the print job ends, whereupon TurboDOS closes the print file and queues it for de-spooling.

In most cases, the print job ends automatically at the conclusion of the program. However, the print job may also be ended by an end-print attention request from the console, by an explicit end-print request from the program, or by the presence of a reserved end-of-print character in the print output stream (if one is defined for your system).

Using Print Queues

TurboDOS supports up to 16 print queues, identified by the letters A-P. A print queue is simply a list of print jobs awaiting de-spooled printing. You can assign any printer on your system to de-spool from any print queue. Jobs are always printed from a particular queue on a first-come first-served basis.

The simplest way to use these queues is to assign each printer to a different queue -- for example, printer A to queue A, printer B to queue B, and so on. However, queues may be used in more imaginative ways.

Using Print Queues (Continued)

Even if your computer has only one printer, you may want to make use of several print queues to group together print jobs with similar forms requirements and/or priorities. For example, you could use queue A for jobs requiring wide paper, queue B for jobs to be printed on narrow paper, queue C for jobs to be printed on pre-printed invoice forms, queue D for computer-printed checks, and so forth. Whenever the printer is done printing all jobs from one queue, you can re-assign the printer to a different queue after changing to the appropriate kind of paper.

If your system has multiple printers, you can assign two or more printers to the same print queue. In this case, TurboDOS performs automatic load sharing by dividing up the workload among the printers. This technique is appropriate if you don't care which printer is used to print which job.

Manual Spooling

In addition to the automatic print spooling already described, TurboDOS supports a manual spooling mode in which print output is spooled to a print file but not automatically queued for printing when the print job ends. When you use this mode, the print file remains on disk indefinitely, until you either delete it or manually queue it for printing. TurboDOS provides a special command for manually queuing such files (or any text file, for that matter).

The manual spooling mode is useful in various situations: for example, when you're not sure whether or not you want to print a job; when you need to print several copies of the same job; or when you want to use the print output of one program as the input to some other program.

De-Spooling

De-spooling is an automatic background activity which generally requires no operator attention. However, TurboDOS lets you exercise control over de-spooling in various ways when necessary.

You can assign any printer to de-spool from any print queue, or place any printer in an "off-line" status (to allow you to change paper or ribbons, for example). You can stop and restart printing on any printer, restart any print job from the beginning, or terminate any print job altogether. These control functions may be exercised from any user console.

Now that we've reviewed the printing capabilities of TurboDOS, let's look at the commands for controlling print routing, queuing, and de-spooling. They are PRINT, QUEUE, and PRINTER.

Routing Control

The PRINT command controls the routing of print output. In a multi-user system, each user may control his own print routing independently.

Direct Printing

To select the direct printing mode, you need only specify which printer (A-P) you wish to use:

```
0A}PRINT PRINTER=B
Printing is to PRINTER B
0A}
```

When using direct printing in a multi-user system, make sure no one else uses that printer until your printing is finished.

Automatic Spooling

To select the automatic spooling mode, you must specify on which queue (A-P) you want your print job placed. You may also specify the disk drive onto which you want the print file to be written:

```
0A}PRINT DRIVE=C QUEUE=A
Printing is to SPOOLER on DRIVE C
             to QUEUE A
0A}PRINT QUEUE=B
Printing is to SPOOLER on DRIVE C
             to QUEUE B
0A}
```

If you don't specify the spool drive, it will remain the same as before.

Manual Spooling

To select the manual spooling mode (in which the print file is not automatically queued), the keyword FILE is used:

```
0A}PRINT FILE
Printing is to SPOOLER on DRIVE C
0A}PRINT FILE DRIVE=A
Printing is to SPOOLER on DRIVE A
0A}
```

Again, you need to specify the spool drive only if you want to change it.

Other Options

You can direct print output to your console with the following command:

```
0A}PRINT CONSOLE
Printing is to CONSOLE
0A}
```

You can cause print output to be discarded altogether by using this command:

```
0A}PRINT OFFLINE
Printing is to OFFLINE
0A}
```

Finally, a PRINT command with no parameters displays your current print routing:

```
0A}PRINT
Printing is to SPOOLER on DRIVE A
             to QUEUE A
0A}
```

Queuing Manually

The QUEUE command lets you manually queue print files (or any text file) for de-spooled printing:

```
0A}QUEUE TEXTFILE.PRN
0A:TEXTFILE.PRN queued
0A}
```

The QUEUE command lets you specify which queue you want to use, but if you don't (as in the example above) the current queue is used (as set by the last PRINT command). You can specify that you want the file to be deleted after it is printed. You can even use wild-cards in the file name to queue a group of files with a single command. For details on all these variations, refer to the QUEUE command description later in this document.

De-Spool Control

The PRINTER command lets you control de-spooled printing on any printer in your system. This command may be used by any user from any console.

Queue Assignment

To assign a printer (A-P) to de-spool from a particular print queue (A-P):

```
0A}PRINTER A QUEUE=C
PRINTER A assigned to QUEUE C
0A}
```

If the printer is currently printing, the new assignment takes effect at the end of the current print job.

To display the current de-spool assignment of a particular printer:

```
0A}PRINTER A
PRINTER A assigned to QUEUE C
0A}
```

To take a specified printer off-line at the end of the current print job:

```
0A}PRINTER A OFFLINE
PRINTER A assigned to OFFLINE
0A}
```

The purpose of taking a printer off-line is to prevent subsequent de-spooling to that printer. This is useful when you want to change paper or ribbons, or when you want to reserve the printer for direct printing.

Print Job Control

To temporarily suspend de-spooling to a specified printer (to correct a paper jam, for example):

```
0A}PRINTER A STOP
PRINTER A assigned to QUEUE C (Stopped)
0A}
```

To resume de-spooling from the point it was stopped:

```
0A}PRINTER A GO
PRINTER A assigned to QUEUE C
0A}
```

To stop de-spooling to a specified printer and restart the current print job from the beginning when de-spooling is resumed:

```
0A}PRINTER A BEGIN
PRINTER A assigned to QUEUE C (Stopped)
0A}
```

To terminate the current print job on a specified printer, and continue with the next queued job:

```
0A}PRINTER A TERMINATE
PRINTER A assigned to QUEUE C
0A}
```

The terminated print file is not deleted from disk, so the job may be manually re-queued with the QUEUE command.

Spooler Errors

If you hear a "beep" and see the following diagnostic message on your console:

```
┌──────────────────────────────────────┐
│                                      │
│   Spooler Error (Ignore, Abort)      │
│                                      │
└──────────────────────────────────────┘
```

it means that the disk to which you were spooling has run out of space in the middle of your print job. TurboDOS has closed your print file (prematurely), but has not queued it for de-spooling.

When you see this message, you must choose one of the two recovery options (Ignore or Abort) by keying the letter I or A.

If you key I (ignore), your print routing is set to "off-line", and your program will continue with any further print output discarded.

If you key A (abort), your program will be terminated.

Summary

In this section, you've learned all about printing and print spooling in all its many variations.

Next, you will find out more about how TurboDOS processes commands.

PROCESSING

This section teaches you about the various processing methods available under TurboDOS, including interactive and batch processing in both single and multi-processor environments.

Command Processing

A TurboDOS command always identifies a program file residing on disk, and causes that program to be loaded into memory and executed. TurboDOS has no "built-in" commands.

TurboDOS comes with more than 30 standard command programs (described in detail in the next section). You can expand the vocabulary of commands by adding programs you purchase or programs you write yourself. Program files are distinguished by their special type: .COM for Z80 programs, .CMD for 8086 programs.

Simple Commands

Every TurboDOS command consists of the file name of the program to be executed, possibly followed by an optional "command tail".

The program name may have an explicit file type, but usually doesn't (TurboDOS assumes .COM or .CMD as appropriate). It may also have an explicit drive specification (like "B:") if the program is not on the current drive.

The format of the command tail is determined by the particular command (program) involved. TurboDOS simply passes the command tail (if any) to the program. The command tail cannot exceed 127 characters in length.

Command Strings

TurboDOS also accepts strings of commands separated by the character \ (backslant). TurboDOS executes each command in sequence, and re-displays each but the first as it is executed:

```
0A}DIR B:\DIR C:\DIR D:

...(drive B directory)...

0A}DIR C:

...(drive C directory)...

0A}DIR D:

...(drive D directory)...

0A}
```

A command string may not exceed the size of the TurboDOS command buffer, which is normally big enough to accomodate a command string two lines long.

Batch Processing

Sometimes you will find yourself performing certain command sequences quite frequently. It's convenient to save such sequences on disk, especially if they're long or complex. You can perform such pre-defined command sequences automatically with the DO command.

Simple Do-Files

Do-files are text files (usually type .DO). You can prepare do-files with your favorite text editing or word processing program. For example:

```
0A}TYPE RUNPROG.DO
BASCOM PROG.REL,PROG.PRN=PROG.BAS/S/C
TYPE PROG.PRN ;L
DELETE PROG.PRN
L80 PROG.REL/M/E,PROG.COM/N
DELETE PROG.REL
PROG
0A}
```

To execute the do-file, use the DO command:

```
0A}DO RUNPROG
0A}BASCOM PROG.REL,PROG.PRN=PROG.BAS/S/C
...(compilation)...
0A}TYPE PROG.PRN ;L
...(listing to printer)...
0A}DELETE PROG.PRN
 0A:PROG    .PRN deleted
0A}L80 PROG.REL/M/E,PROG.COM/N
...(link map)...
0A}DELETE PROG.REL
 0A:PROG    .REL deleted
0A}PROG
...(execution of PROG program)...
0A}
```

Simple Do-Files (Continued)

Certain commands (such as COPY, RENAME and DELETE) may expect interactive input from the console keyboard. If such a command is executed from within a do-file, then its console input comes from the do-file rather than the keyboard. For example:

```
0A}TYPE DELPROG.DO
DELETE
PROG.PRN
PROG.REL
PROG.COM
0A}DO DELPROG
0A}DELETE
* PROG.PRN
 0A:PROG.PRN deleted
* PROG.REL
 0A:PROG.REL deleted
* PROG.COM
 0A:PROG.COM deleted
*
0A}
```

Most programs which expect keyboard input may be run from within a do-file in this fashion.

Do-File Parameters

You can prepare a do-file with variable parameters which can have different values each time you execute it. When you prepare the do-file, mark the position of parameters in the do-file as {1}, {2}, and so on:

```
0A}TYPE RUNBAS.DO
BASCOM {1}.REL,{1}.PRN={1}.BAS/S/C
TYPE {1}.PRN ;L
DELETE {1}.PRN
L80 {1}.REL{2}/E,{1}.COM/N
DELETE {1}.REL
{1}
0A}
```

When you execute the do file, specify the argument values in the command tail of the DO command. The first argument replaces each occurrence of {1} in the do-file, the second argument replaces each {2}, and so forth:

```
0A}DO RUNBAS PROG /M
0A}BASCOM PROG.REL,PROG.PRN=PROG.BAS/S/C
...(compilation)...
0A}TYPE PROG.PRN ;L
...(listing to printer)...
0A}DELETE PROG.PRN
 0A:PROG    .PRN deleted
0A}L80 PROG.REL/M/E,PROG.COM/N
...(link map)...
0A}DELETE PROG.REL
 0A:PROG    .REL deleted
0A}PROG
...(execution of PROG program)...
0A}DELETE RUNBAS.DO$
 0A:RUNBAS  .DO$ deleted
0A}
```

Do-File Parameters (Continued)

When arguments are present, the DO command makes a temporary copy of your do-file in which arguments are substituted for parameters as required. The temporary file is executed, and then deleted when execution is done.

Nested Do-Files

Do-files may contain embedded DO commands:

```
0A}TYPE RUN.DO
DO COMPILE {1}
DO LINK {1} {2}
{1}
0A}TYPE COMPILE.DO
BASCOM {1}.REL,{1}.PRN={1}.BAS/S/C
TYPE {1}.PRN ;L
DELETE {1}.PRN
0A}TYPE LINK.DO
L80 {1}.REL{2}/E,{1}.COM/N
DELETE {1}.REL
0A}
```

Do-files may nested in this fashion to any reasonable depth.

Automatic Loading

TurboDOS provides a simple means for automatically loading any program or executing any sequence of commands at initial start-up (cold start), or whenever a program terminates (warm start). You could use this feature to cause automatic execution of an application function menu program or the LOGON command, for example.

Autoload at cold-start takes place only if a file named COLDSTRT.AUT is present on the start-up disk. Autoload at warm-start takes place only if a file named WARMSTRT.AUT is present on the current disk.

In NorthStar TurboDOS, the UP8, UP16, and Background Batch-UP8 operating systems have been configured to look for different warmstart files, so that all warmstart files cna reside in User 31. The UP8 looks for WRM8STRT.AUT; the UP16 looks for WRM6STRT.AUT; and the Background Batch looks for WRMBSTRT.AUT.

Program Autoload

To cause a program or command to be loaded automatically, simply make a copy of the .COM or .CMD file under the name COLDSTRT.AUT or the appropriate WRMxSTRT.AUT. For instance, for automatic UP8 LOGON operation, copy LOGON.COM to WRM8STRT.AUT under user 31.

Command Autoload

To cause a command or string of commands to be executed automatically, use the AUTOLOAD command to capture the desired command string in a file called AUTOLOAD.AUT. Then rename this file to COLDSTRT.AUT or WARMSTRT.AUT as appropriate. For example, to automatically set the system date and then execute a BASIC program called MENU at each cold-start:

```
0A}AUTOLOAD DATE SET|BASIC MENU
Autoload file created.
```

(cont.)

```
0A}RENAME AUTOLOAD.AUT COLDSTRT.AUT
 0A:AUTOLOAD.AUT renamed 0A:COLDSTRT.AUT
0A}
```

Note that a newly-created .AUT file does not take effect until the next cold-start.

Networking

As you know already, a multi-user TurboDOS system is actually a network of interconnected microcomputers working together. A separate microcomputer supports each user's video console, and one or more additional microcomputers may be used to manage disks, printers, communications channels, and other shared devices of the system.

In general, you do not need to be concerned about the networking aspects of TurboDOS. When you read from drive C or print to printer B, for example, you do not need to think about whether that disk drive is connected to your processor or to another processor in the network. TurboDOS automatically routes your requests to the proper destination without any explicit action on your part.

However, there are a few cases in which you need to be conscious of the true multi-processor nature of the system.

Local Commands

A few TurboDOS commands require direct access to disk drive and controller hardware, and consequently may be executed only in the processor to which the disk is attached. These commands are:

```
BACKUP - Copy entire disk track-by-track
BOOT   - Read/write reserved tracks
FORMAT - Erase and initialize entire disk
VERIFY - Scan entire disk for bad spots
```

If you try to execute any of these commands in another processor, you will get the error message "Not ready error, Drive X".

Remote Console

In many TurboDOS systems, disks and other shared peripherals are attached to a "master" or "server" processor which is separate from the processors to which user consoles are attached. If you need to run a program in the server processor (e.g., FORMAT), you would ordinarily need to hook up a console device to the server. However, TurboDOS has a special command (SERVER) which allows you to temporarily assign any user console to the server processor without making any hardware changes:

```
5C}SERVER
Console attached to server processor
0A}FORMAT B:
      :
      :
0A}[BREAK] [CONTROL-C]
Console detached from server processor
5C}
```

The SERVER command attaches your console to the server processor. To detach, key in the BREAK sequence.

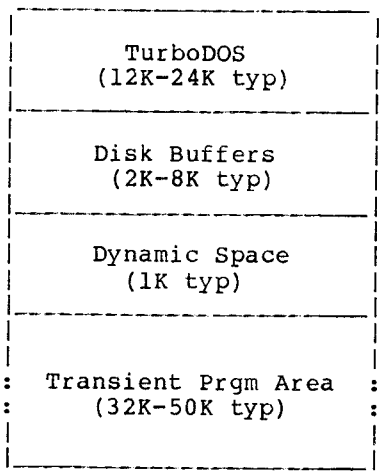
Memory Management

Microcomputers use random-access memory (RAM) to hold both programs and data. TurboDOS itself occupies a certain amount of memory, and it needs additional memory for disk buffers, print queues, file interlocks, and various other things. All the memory space that is left over is called the "Transient Program Area" (TPA), and is available for use by commands and application programs.

Because memory is often a critical resource, it is important for you to understand how it is managed by TurboDOS. This varies depending upon the hardware involved.

Non-Banked Z80

A non-banked Z80 configuration is limited by hardware constraints to a maximum of 64K of memory. TurboDOS resides in the topmost portion of memory, and allocates its disk buffers and other dynamic space requirements immediately below itself. The TPA occupies the lower portion of memory:

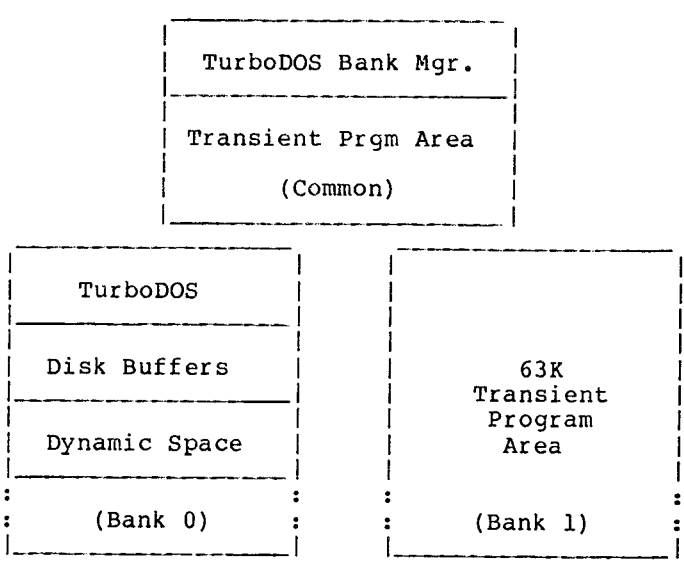


If you need additional TPA for a particular program, you can reduce the size of the disk buffer pool by using the BUFFERS command. However, you must make sure the system is quiescent (no programs running, no files open, no print jobs queued, etc.) so that there is no dynamic space allocated below the disk buffers. Then you can reduce the disk buffers and increase your TPA accordingly.

Banked Z80

A banked Z80 configuration permits more than 64K of memory by providing two memory banks (called "bank 0" and "bank 1"), only one of which may be active at a time. TurboDOS resides in bank 0 along with its disk buffers and other dynamic space, while the TPA occupies bank 1.

In order to allow the necessary inter-bank communications to take place, an area of common (non-switched) memory must be provided at the top of memory. Ideally, this common area should be 1K, permitting each memory bank to be 63K. However, hardware design often dictates that the common area be larger than this, often as big as 16K. A large common area reduces the memory available for TurboDOS, but does not reduce the TPA size:

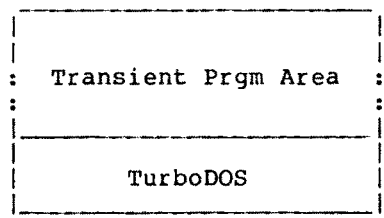


Banked Z80 (Continued)

Certain TurboDOS commands (BACKUP, BOOT, FORMAT, and VERIFY) will not run in a banked TPA. So that you can use these commands, TurboDOS provides a BANK command which lets you change between the banked and non-banked configurations at will.

8086-Family

In an 8086-family processor, TurboDOS occupies lower memory, with the TPA above:



Since 8086-family processors can address up to a megabyte of memory directly, memory management is generally not a problem, and bank-switching is unnecessary.

Summary

In this section, you've learned how TurboDOS commands work, and various ways commands may be executed (command strings, do-files, auto-load). You also understand the effects of networking and memory management on your processing.

Now we are ready to examine each TurboDOS command in detail.

COMMANDS

This section describes each TurboDOS command in detail. For ease of reference, the commands are presented in alphabetical order.

Presentation

Each command description contains:

- Purpose
- Syntax
- Explanation
- Options
- Examples
- Error Messages
- Patch Points

Command Syntax

The following notation is used to describe the syntax of commands:

- Keywords are shown in capital letters, and must be entered as shown. However, you can enter them in either upper- or lower-case.
- Parameters are shown in lower-case, and are described in the following text.
- Items shown in braces {} are optional. If you want to include such an optional item, do not type the braces, but only the information inside.
- An ellipsis ... indicates that the preceding item may be repeated as many times as you like.
- You must enter spaces and punctuation characters as shown (except for braces {} and elipses ...).
- Command options are usually introduced by semicolon; several options may be combined following a single semicolon.

AUTOLOAD Command

The AUTOLOAD command lets you set up a command sequence to be executed automatically at each cold-start or warm-start.

Syntax

```
AUTOLOAD command {|command}...
```

Explanation

If you wish, TurboDOS will automatically load any program or execute any command sequence at initial start-up (cold-start), or at the conclusion of each program you run (warm-start). If the file COLDSTRT.AUT exists on the start-up disk, TurboDOS will execute it automatically at each cold-start. If the file WRMxSTRT.AUT (where x = 8 for UP8, 6 for UP16, B for background batch) exists on the current disk, TurboDOS will execute it automatically at each warm-start.

You can use the AUTOLOAD command to create these .AUT files. The command tail consists of the command(s) to be executed automatically. Note that if multiple commands are given, they must be separated with the delimiter | rather than \.

The AUTOLOAD command always creates a file named AUTOLOAD.AUT. By renaming this file as COLDSTRT.AUT or WARMSTRT.AUT, you will cause the specified command(s) to be executed automatically at each cold-start or warm-start. A newly created .AUT file does not affect system operation until the next system start-up.

Example

```
0A}AUTOLOAD DATE SET|BASIC MENU
Auto load file created
0A}RENAME AUTOLOAD.AUT COLDSTRT.AUT
 0A:AUTOLOAD.AUT renamed 0A:COLDSTRT.AUT
0A}
```

Error Messages

```
Error creating auto load file
```

Patch Points

Z80	8086	Val	Description
103H	CS+4	"|"	Special delimiter
104H	CS+5	"\"	Normal delimiter

BACKUP Command

The BACKUP command performs a fast copy of an entire disk onto another disk of the same type and format, and is generally used for purposes of backup.

Syntax

```
BACKUP s: d: {;options}
```

Explanation

The BACKUP performs a track-by-track copy from the source drive "s:" to the destination drive "d:". Source and destination disks must be of exactly the same type and format, and must both be attached to the processor in which the command is executed. Use of this command is restricted to privileged log-ons only.

Options

Option	Explanation
;R	BACKUP repeats continuously (allowing multiple disks to be backed up) until terminated by typing CTRL-C.

Example

```
0A}BACKUP A: B:
Insert source disk in drive A
Insert destination disk in drive B
Enter <CR> to begin copying [RETURN]
.....................................
Successful copy
0A}
```

Error Messages

```
Non-privileged user
No source drive specified
No destination drive specified
Unable to lockout source drive
Unable to lockout destination drive
Source drive not ready
Destination drive not ready
Disk types not identical
Insufficient memory for copy
Read error, Drive: d, Track: nn
Write error, Drive: d, Track: nn
Unable to execute from bank 1
```

BANK Command

In a banked Z80 system, the BANK command lets you to change from bank-switched to non-bank-switched operation, and vice versa.

Syntax

```
BANK 0
BANK 1
BANK
```

Explanation

The first form selects the non-bank-switched mode of operation in which the TPA resides in memory bank 0 (along with TurboDOS). The second form selects the bank-switched mode in which the TPA occupies bank 1. The third form simply displays the current bank mode.

The BANK command has no effect in a non-banked system, and always shows that bank 0 is selected. Use of this command is restricted to privileged log-ons only.

Examples

```
0A}BANK
Current bank number: 1
0A}BANK 0
Current bank number: 0
0A}
```

Error Messages

```
Invalid bank number requested
Non-privileged user
```

BATCH Command

The BATCH command provides a convenient way of entering TurboDOS command strings into a FIFO for processing by a dedicated batch processor in a networking system.

Syntax

```
BATCH command {|command}...
```

Explanation

To make use of the BATCH command, you must have previously set up a FIFO with the standard name BATCH.DO; in most cases, this FIFO should be set up as a global file under user number 0 on drive A, and should have the "suspend" option selected (see FIFO command). The batch processor executes the command "DO BATCH" (usually autoloaded) at system start-up. Thereafter, any command string written to the FIFO will wake up the batch processor and be executed. Whenever the FIFO becomes empty, the batch processor will be suspended awaiting additional work to do.

You can use the BATCH command to write any command or string of commands to the FIFO. If multiple commands are given in the BATCH command tail, they must be separated with the delimiter | rather than \. BATCH writes the command(s) to BATCH.DO, converting each | to a \ and prefixing the command string with commands to select the proper drive and user number.

Example

```
5C}BATCH PASM PROG|RELCVT PROG
Message sent to FIFO
5C}
```

The command above writes the following text to BATCH.DO:

```
USER 5\C:\PASM PROG\RELCVT PROG
```

Error Messages

```
BATCH.DO FIFO not found
BATCH.DO file not FIFO
BATCH.DO FIFO is full
Excessive command length
```

Patch Points

Z80	8086	Val	Description
103H	CS+4	"|"	Special delimiter
104H	CS+5	"\"	Normal delimiter
105H	CS+6	0	Drive for BATCH.DO (0=A, 1=B,..., 15=P)

BOOT Command

The BOOT command lets you access any reserved tracks ("boot tracks") on a disk.

Syntax

```
BOOT s: destfile
BOOT srcefile d:
BOOT s: d:
```

Explanation

Some hardware configurations require that certain tracks on a disk be reserved (e.g., for a cold-start loader or alternate track table). The BOOT command provides a means for reading, writing, or copying such reserved tracks.

The first form reads the reserved tracks from the source drive "s:" and writes the data to the destination file "destfile". The second form reads a source file "srcefile" and writes the contents to the reserved tracks on destination drive "d:". The last form copies the reserved tracks from source drive "s:" to destination drive "d:".

The BOOT command reads the entire source (tracks or file) into memory before writing the data out to the destination (tracks or file). If the source is too big to fit into the TPA, you will get an error message. Use of the BOOT command is restricted to privileged log-ons only.

Examples

```
0A}BOOT B: BOOTSAVE.SYS
Reading boot tracks: ...
Writing destination file: .........
Operation successful
0A}BOOT BOOTCODE.SYS B:
Reading source file: ..........
Writing boot tracks: ...
Operation successful
0A}BOOT B: A:
Reading boot tracks: ...
Writing boot tracks: ...
Operation successful
0A}
```

Error Messages

```
Non-privileged user
No source drive/filename specified
No destination drive/filename specified
Can't find source file
Can't read boot tracks
Can't write destination file
Can't write boot tracks
Not enough memory
Unable to execute from bank 1
```

BUFFERS Command

The BUFFERS command lets you change the number and/or size of disk buffers maintained by TurboDOS.

Syntax

```
BUFFERS {Nnumber} {Ssize}
```

Explanation

The BUFFERS command sets the number of buffers to "number" and the buffer size to "size". The minimum number of buffers allowed by TurboDOS is 2. The buffer size is given in decimal bytes, and must be one of the following: 128, 256, 512, 1024, 2048, 4096, 8192, or 16384. The buffer size should not be smaller than the physical sector size of the disks being used.

If there is not enough memory to allocate the requested number of buffers of the specified size, the BUFFERS command will allocate as many as it can. If either the "N" or "S" argument is omitted, the corresponding parameter remains unchanged. If both are omitted, the command simply displays the current parameters.

The use of the BUFFERS command to change buffer parameters is restricted to privileged log-ons only. In a networking system, if the BUFFERS command is executed in a slave processor without any local disk storage, then it refers to the buffer pool in the master processor.

Example

```
0A}BUFFERS N6
Number of buffers: 6
Length of buffers: 1024
0A}
```

Error Messages

```
Non-privileged user
Minimum number of buffers = 2
Maximum number of buffers = 255
Invalid buffer size requested
```

CHANGE Command

You must use the CHANGE command prior to removing a disk from any drive on a multi-user TurboDOS system.

Syntax

```
CHANGE {drivelist}
```

Explanation

The command tail consists of a list of drive letters A...P corresponding to the disks that you want to change. You may enter an asterisk * if you want to change all disks. If you omit the drive list, CHANGE will prompt you for it.

If any of the drives you request are in use by another user, your request will be denied. Otherwise, you will be prompted to change the requested disk(s), and to enter a carriage-return when you are done. Until you have pressed carriage-return, no other user will be allowed to access the disk(s) that you are changing.

Before shutting down a multi-user TurboDOS system, it is a good idea to enter the command "CHANGE *".

Examples

```
0A}CHANGE BCD
Change drive(s) BCD
Enter <CR> to configure [RETURN]
0A}CHANGE *
Following drive(s) in use: CD
0A}
```

Error Messages

```
Invalid drive(s) requested
Following drive(s) in use: d...
Unable to free drive(s): d...
```

COPY Command

The COPY command lets you copy individual disk files or groups of files.

Syntax

```
COPY srcefile destfile {;options}
COPY {;options}
```

Explanation

The COPY command makes a copy of the file specified by "srcefile". The new file is created on the drive and with the filename specified by "destfile".

You may use wild-card characters (? and *) in the "srcefile" argument to indicate that multiple files are to be copied. Wild-card characters may also be used in the "destfile" argument to indicate that the corresponding characters of each source filename are to be used in the destination filename. If you give a "srcefile" or "destfile" argument consisting of only a drive letter, COPY assumes a filename of all wild cards (in other words, C: is equivalent to C:*.*).

If you omit both "srcefile" and "destfile" from the command (second form), then the COPY command operates in an interactive mode. You are prompted by an asterisk * to enter a series of directives from the console. The syntax of each directive is:

```
srcefile destfile {;options}
```

Options specified within a directive take precedence over options specified in the COPY command itself. A null directive (RETURN only) terminates the COPY command.

Options

Option	Explanation
;A	Only source files without the "archived" attribute are copied. These source files are given the "archived" attribute after they are copied. This provides a means for incremental backup of large disks.
;B	Allows a large source file to be copied in sections to multiple destination disks, or to restore such a file from multiple source disks. If this option is specified, the "srcefile" argument cannot contain wild cards.
;C	If the destination fills up, COPY allows you to change the disk in the destination drive and continue copying. This is useful when backing up a hard disk to floppy disks.
;Dnn	COPY creates destination files under user number nn. This option is honored for privileged log-ons only.
;E	COPY erases each source file after copying to destination.
;N	If "srcefile" contains wild cards, COPY does not ask you for confirmation before copying each file. If a destination file already exists, COPY does not ask you for confirmation before deleting it.

Options (Continued)

Option	Explanation
;R	COPY replaces files on the destination disk, but doesn't copy files that don't already exist on the destination disk
;Snn	COPY uses source files under user number nn. This option is honored for privileged log-ons only.
;X	Copying is not performed if the destination file already exists.
;Y	If "srcefile" contains wild cards, COPY asks you for confirmation before copying each file. If a destination file already exists, COPY asks you for confirmation before deleting it.

Examples

```
0A}COPY *.BAS B: ;N
0A:AMORTIZE.BAS copied to 0B:AMORTIZE.BAS
0A:PRIMES  .BAS copied to 0B:PRIMES  .BAS
0B:STARTREK.BAS already exists, deleted
0A:STARTREK.BAS copied to 0B:STARTREK.BAS
0A}COPY
* AUTOLOAD.COM B: ;E
0A:AUTOLOAD.COM copied to 0B:AUTOLOAD.COM
0A:AUTOLOAD.COM deleted
* [RETURN]
0A}
```

Error Messages

```
Invalid filename 1
Invalid filename 2
Ambiguous filename not permitted with
     option B
Destination drive read-only
<filename> copy to same filename
<filename> FIFO file, not copied
File not found
Unable to lock destination drive
Unable to free drive
Unable to open source file
Unable to create destination file
Unable to read source file
Unable to write destination file
Unable to close source file
Unable to close destination file
Unable to set attributes on source file
Unable to set attributes on destination
     file
Insufficient memory
Non-privileged user
```

DATE Command

The DATE command lets you set or display the system date and time.

Syntax

```
DATE {SET}
```

Explanation

If "SET" is specified, the DATE command prompts you interactively to enter the new system date and time. The required format for the date is "dd mmm yy", and for the time "hh:mm:ss" (see Examples). You may leave the date or time unchanged by typing RETURN in response to the prompt.

If "SET" is not specified, the DATE command simply displays the current system date and time.

Examples

```
0A}DATE SET
Date: 07 Dec 41
Time: 16:15:00
0A}DATE
Date: 07 Dec 41
Time: 16:15:08
0A}
```

DELETE Command

The DELETE command lets you delete individual disk files or groups of files.

Syntax

```
DELETE filename {;options}
DELETE {;options}
```

Explanation

The DELETE command permanently deletes the file specified by "filename" from disk.

You may use wild-card characters (? and *) in the "filename" argument to indicate that multiple files are to be deleted.

If you omit "filename" from the command (second form), then the DELETE command operates in an interactive mode. You are prompted by an asterisk * to enter a series of directives from the console. The syntax of each directive is:

```
filename {;options}
```

Options specified within a directive take precedence over options specified in the DELETE command itself. A null directive (RETURN only) terminates the DELETE command.

Options

Option	Explanation
;N	If "filename" contains wild cards, DELETE does not ask you for confirmation before deleting each file.
;Y	If "filename" contains wild cards, DELETE asks you for confirmation before deleting each file.

Examples

```
0A}DELETE *.BAS ;N
 0A:AMORTIZE.BAS deleted
 0A:PRIMES  .BAS deleted
 0A:STARTREK.BAS deleted
0A}DELETE B:MAXI*.* ;Y
 0B:MAXICOMP.TXT   OK to delete (y/n)? N
 0B:MAXIMUMS.COM   OK to delete (y/n)? Y
 0B:MAXIMUMS.COM deleted
 0B:MAXIMUMS.ASM   OK to delete (y/n)? Y
 0B:MAXIMUMS.ASM deleted
0A}DELETE
* AUTOLOAD.COM
 0A:AUTOLOAD.COM deleted
* *.TXT ;N
 0A:REFERENC.TXT deleted
 0A:USERGUID.TXT deleted
* [RETURN]
0A}
```

Error Messages

```
Destination drive read-only
<filename> read-only, not deleted
<filename> FIFO file, not deleted
File not found
Invalid filename
Insufficient memory
Network error
```

DIR Command

The DIR command displays an alphabetized disk directory on the console or printer.

Syntax

```
DIR filename {;options}
DIR d: {;options}
DIR {;options}
```

Explanation

The "filename" argument generally contains wild-card characters (? and *), and specifies the drive and group of files to be included in the directory. If you specify only a drive letter (second form), all files on the specified drive are displayed. If you omit the drive letter as well (third form), all files on the current drive are displayed.

In all cases, the directory display starts out with a preamble containing the following information:

- user number, drive, and "filename"
- disk label
- current date and time
- hashed directory indicator: (H)
- free space remaining on disk
- number of files displayed
- combined size of files displayed

This preamble is followed by an alphabetized list of filenames, with the size of each file shown. Read-only files are distinguished by a colon : (rather than a period .) between the filename and filetype.

If there are too many files to fit on the screen, DIR waits at the end of each screen-full for you to press RETURN.

Options

Option	Explanation
;L	The directory is printed rather than displayed on the console.
;Unn	The directory for user number nn is displayed. This option is honored for privileged log-ons only.

Examples

```
0A}DIR ;L
...(printout of all 0A: files)...
0A}DIR B: ;U5
...(display of all 5A: files)...
0A}DIR *.BAS
...(display of only 0A:*.BAS files)...
0A}
```

Error Messages

```
Insufficient memory
Network error
```

Patch Points

Z80	8086	Val	Description
103H	CS+4	3	Left margin for ;L opt
104H	CS+5	"^L"	Clear-screen character

DO Command

The DO command lets you execute a pre-defined sequence of TurboDOS commands which you have previously saved in a disk file.

Syntax

```
DO filename {arg1 arg2 ... argN}
```

Explanation

The "filename" argument specifies the file of TurboDOS commands to be executed (called the "do-file"). If "filename" does not specify an explicit filetype, then .DO is assumed. The DO command causes the commands contained in the do-file to be executed in sequence. A do-file may contain embedded DO commands, and TurboDOS supports such "nesting" of do-files to any reasonable depth.

The optional arguments "arg1" through "argN" are substituted into marked locations in the do-file. Any number of arguments is permitted. If arguments contain embedded spaces, they must be enclosed in single or double quotes. The presence of one or more of these arguments causes the DO command to make a temporary copy of the do-file in which the arguments are substituted as required. The commands in the temporary file are then executed in sequence. The temporary file is given the same filename as the original do-file, except that the last character of the filetype is changed to a dollar sign (e.g., GENERATE.DO is copied to GENERATE.DO$). The last entry in the temporary file is a DELETE command which causes the temporary file to be deleted.

The do-file is simply a text file, each line of which contains a valid TurboDOS command or command string. You can create do-files with any text editing program.

Explanation (Continued)

If argument substitution is desired, then you must mark each substitution point in the do-file by enclosing the argument number in braces {}. For example, {3} in the do-file will be replaced by the value of "arg3" of the DO command. A default value may follow the argument number, separated by a comma ({3,TEMP} for example), and will be used if the corresponding argument of the DO command is missing or null.

Certain commands (such as COPY, RENAME and DELETE) and other programs expect interactive input from the console. If such a command or program is executed within a do-file, then its console input comes from the do-file rather than the console (in most cases).

Examples

See preceding section on Batch Processing for examples of the DO command.

Error Messages

```
Unable to activate DO file
```

Patch Points

Z80	8086	Val	Description
103H	CS+4	"{"	Left param delimiter
104H	CS+5	"}"	Right param delimiter

DRIVE Command

The DRIVE command displays information about the format of a disk.

Syntax

```
DRIVE {d:} {;options}
```

Explanation

The DRIVE command displays the format of the disk specified by the "d:" argument, as shown in the example below. If you omit the "d:" argument, then the format of the disk in the current drive is displayed. The display may be on the console or printer.

Options

Option	Explanation
;L	The disk format information is printed, rather than displayed on the console.

Example

```
0A}DRIVE B:
 Disk format, drive B:DOCUMENT.TXT
 Maximum data capacity      : 1224K
 Allocation block size      : 2K
 Number of directory entries: 1024
 Physical sector size       : 1024
 Physical sectors per track : 16
 Physical tracks per disk   : 77
 Number of reserved tracks  : 0
 Removable media
0A}
```

Error Messages

```
Network error
```

DUMP Command

The DUMP command displays a combined hexadecimal and ASCII file dump on the console or printer.

Syntax

```
DUMP filename {;options}
```

Explanation

The DUMP command displays the contents of the file specified by "filename" in both hexadecimal and ASCII formats. The dump may be directed to either the console or printer.

Options

Option	Explanation
;L	The dump is printed, rather than displayed on the console.

Example

```
0A}DUMP B:DUMP.COM
...(hex/ASCII dump of B:DUMP.COM)...
0A}
```

Error Messages

```
File not found
```

ERASEDIR Command

The ERASEDIR command lets you erase the entire directory of a disk.

Syntax

```
ERASEDIR d:
```

Explanation

The ERASEDIR command erases all files on the disk specified by the "d:" argument, regardless of user number or read-only attributes. It also lets you specify whether the newly-initialized directory should be maintained in linear or hashed format.

Use of the ERASEDIR command is restricted to privileged log-ons only.

Example

```
0A}ERASEDIR B:
Hashed directory desired (Y/N)? Y
OK to erase directory on drive B (Y/N)? Y
Directory erased, hashed
0A}
```

Error Messages

```
Non-privileged user
No drive specified
Unable to erase directory
```

FIFO Command

The FIFO command lets you create FIFO files.

Syntax

```
FIFO filename
```

Explanation

FIFO's are special files supported by TurboDOS to facilitate communications between users and processors. More details may be found in the TurboDOS Programmers Guide.

If the FIFO identified by "filename" already exists, then the FIFO command displays its characteristics. Otherwise, the FIFO command prompts you interactively for the necessary parameters to create a new FIFO.

Examples

```
0A}FIFO B:BATCH.DO
FIFO file not found, creating new file
Enter FIFO type (Ram/Disk): D
Suspend processing on full/empty (Y/N): Y
Enter max nbr of records (1-65535): 1000
FIFO file created
0A}FIFO B:BATCH.DO
FIFO is Disk resident
FIFO does suspend process on full/empty
Maximum number of records: 1000
Current number of records: 0
0A}
```

Error Messages

```
No FIFO file name specified
File not FIFO
Unable to create FIFO file
Unable to read FIFO file header
```

FIXDIR Command

The FIXDIR command lets you reorganize a disk directory whenever frequent additions and deletions of files have caused directory access to become less efficient. FIXDIR also lets you convert a disk directory from linear to hashed or vice-versa.

Syntax

```
FIXDIR d:
```

Explanation

The FIXDIR command reorganizes the directory on the disk specified by the "d:" argument. It also lets you specify whether the newly-reorganized directory should be maintained in linear or hashed format.

For linear directories, FIXDIR eliminates all deleted entries and compresses all directory entries at the beginning of the directory area. For hashed directories, FIXDIR re-hashes each entry to ensure optimum access efficiency.

WARNING: Execution of FIXDIR cannot be interrupted by an "attention" from the console. Any attempt to abort via hardware reset is likely to result in loss of one or more files. If conversion from linear to hashed format (or vice-versa) is requested, FIXDIR may take a long time (in extreme cases, hours). Do not attempt conversion unless adequate time is available to complete the operation.

You cannot use FIXDIR on a drive that is in use by another user. Use of the FIXDIR command is restricted to privileged log-ons only.

Example

```
0A}FIXDIR B:
Hashed directory desired (Y/N)? Y
OK to reorganize directory on drive B? Y
Reorganizing directory
Directory reorganized, hashed
0A}
```

Error Messages

```
Non-privileged user
No drive specified
Unable to reorganize directory
```

FIXMAP Command

The FIXMAP command lets you regenerate the allocation map for a disk.

Syntax

```
FIXMAP d:
```

Explanation

In order to keep track of which blocks of disk space are occupied and which are free, TurboDOS maintains on each disk an allocation map for that disk's space. Certain program malfunctions (failure to close a newly-created file, for example) can create discrepancies between the allocation map and the directory of a disk. The result is that disk blocks may occasionally become unavailable.

The FIXMAP command regenerates the allocation map on the disk specified by the "d:" argument, thereby reclaiming any disk blocks that may have become unavailable in this fashion.

You cannot use FIXMAP on a drive that is in use by another user. Use of the FIXMAP command is restricted to privileged log-ons only.

Examples

```
0A}FIXMAP B:
OK to regenerate disk map on drive B? Y
Regenerating disk map
Disk map regenerated
0A}FIXMAP C:
OK to regenerate disk map on drive C? Y
Regenerating disk map
Disk map regenerated, 3 blocks gained
0A}
```

Error Messages

```
Unable to regenerate disk map
Non-privileged user
No drive specified
```

FORMAT Command

You must use the FORMAT command to pre-record format information on each new disk before it is used for the first time.

Syntax

```
FORMAT d: {;options}
```

Explanation

The FORMAT command initializes and then verifies the disk identified by "d:", which must be attached to the processor in which the command is executed. Use of the FORMAT command is restricted to privileged log-ons only.

If the hardware supports multiple disk formats, the FORMAT command may require certain hardware-dependent options to specify the particular format desired. If you don't include all of the necessary options in the command, then FORMAT will prompt you for them interactively. The examples below are typical for floppy disks.

Options

Option	Explanation
;R	FORMAT repeats continuously (allowing multiple disks to be formatted) until terminated by typing CTRL-C.
;V	FORMAT performs only the verify pass (and omits the formatting pass).

Options (Continued)

Option	Explanation
;1	CP/M Double-Density (Single-Sided)
;2	CP/M Quad-Capacity (Double-Sided)
;3	TurboDOS Double-Density (Single-Sided)
;4	TurboDOS Quad-Capacity (Double-Sided)

Example

```
0A}FORMAT N:
Enter Format Type:
1=NorthStar CP/M Double-Density
            (Single-Sided)
2=NorthStar CP/M Quad-Capacity
            (Double-Sided)
3=NorthStar TurboDOS Double-Density
            (Single-Sided)
4=NorthStar TurboDOS Quad-Capacity
            (Double-Sided)
=>2
Insert disk to be formatted in drive N
Enter <CR> to begin formatting: [RETURN]
Starting format pass:
.....
Successful format
Starting verify pass:
.....
Successful verify
0A}
```

Error Messages

```
Non-privileged user
Unable to execute from bank 1
No format drive specified
Unable to lockout format drive
Format drive not ready
Verify drive not ready
Insufficient memory to format
Insufficient memory to verify
Format error, Drive: d, Track nn
Verify error, Drive: d, Track nn
```

LABEL Command

The LABEL command lets put a volume label on a disk.

Syntax

```
LABEL filename
```

Explanation

The LABEL command writes a new volume label on the disk identified in the "filename" argument (or the current drive if no drive letter is specified explicitly). The name and type fields of "filename" are used as the volume label. The volume label may be displayed using the DIR or DRIVE commands.

Example

```
0A}LABEL B:PAYABLES.DAT
Disk label written
0A}DRIVE B:
Disk format, drive B:PAYABLES.DAT
...(etc.)...
0A}
```

Error Messages

```
Unable to label disk
Invalid disk label
```

LOGOFF Command

In a multi-user system, the LOGOFF command lets you terminate your session.

Syntax

```
LOGOFF
```

Note: The Turbo-Plus LOGOFF command will normally be used for North Star TurboDOS 8-bit systems. See the Turbo-Plus User's Guide for a description of the Turbo-Plus LOGOFF command.

Explanation

Executing the LOGOFF command has the following effects:

- current user number is set to 31
- current drive is set to system disk
- access to global files is inhibited

The library for user number 31 normally contains only the LOGON command file and the USERID.SYS validation file (see LOGON command). Consequently, no further activity can be performed until a successful LOGON has taken place.

If the file SYSLOG.SYS is also present in the user 31 library, then LOGOFF automatically records the log-off in that file.

Examples

```
5C}LOGOFF
31A}
```

Error Messages

```
Unable to log off
```

LOGON Command

In a multi-user system, the LOGON command lets you start a new console session.

Syntax

```
LOGON
```

Note: The Turbo-Plus LOGON command will normally be used for North Star TurboDOS 8-bit systems. See the Turbo-Plus User's Guide for a description of the Turbo-Plus LOGON command.

Explanation

The LOGON command provides password-type security for the purpose of protecting private file libraries and preventing unauthorized access to the system.

The LOGON command prompts you interactively to enter your user-id, and validates whatever you enter against a validation file called USERID.SYS in the user 31 library. This file is an ordinary text file which may be created with any text editing program, and contains an entry of the following format for each valid user-id:

```
userid, {password}, nn{P}, {d:}
```

where "userid" and "password" are up to eight characters long, "nn" is a user number between 0 and 30, and "d:" is a drive letter. If your user-id has an associated password specified in USERID.SYS, then the LOGON command prompts you to enter a password, and validates it.

If your user-id and password are both found to be valid, then the log-on succeeds. A successful log-on has the following effects:

- current drive is set to "d:"
- current user number is set to "nn"
- access to global files is enabled
- "P" suffix makes you a privileged user

Explanation (Continued)

If the system date and time have not been set since start-up, LOGON prompts for and sets the date and time.

If the file SYSLOG.SYS is present in the user 31 library, then the LOGON command automatically records your log-on in that file. In this case, LOGON prompts you for an activity description, and records it in the log entry.

It is common practice to make a copy of the LOGON command under the name WARMSTRT.AUT in the user 31 library, which causes LOGON to be executed automatically at start-up and immediately after each LOGOFF.

Example

```
31A}LOGON
System log on
Enter user id: Barbara
Enter password: Shazam
Enter date: 15 Apr 84
Enter time: 14:25:30
Enter activity: Payroll
5C}
```

Error Messages

```
Invalid user id
Incorrect password
Invalid date
Unable to log on
```

Patch Points

Z80	8086	Val	Description
103H	CS+4	"^L"	Clear-screen character

MONITOR Command The MONITOR command provides various facilities useful in debugging and patching of programs and files. The command format is:

MONITOR

The MONITOR command operates in an interactive mode. It reads successive directives from the console (prompted by an asterisk "**"). A "Q" directive terminates the command. All addresses and other numeric values are in hexadecimal. The supported directives are:

C val1,val2 calculates the sum and difference between the two specified values.

D addr1,addr2 dumps the area of memory between the two specified addresses. Pause with space, resume with carriage return.

E addr examines memory starting at the specified address. You may substitute a new value for each displayed byte of memory. Enter space or carriage-return to go on to the next byte, or ESC to terminate examine mode.

F addr1,add2[,VAL[,REP]] fills the area of memory between the two specified addresses with the specified value (or zero if no value is given). If the repetition factor is given, the operation is repeated that many times (useful with some EPROM programmers).

G addr goes to the instruction at the specified address.

H displays a "help" menu listing all directives.

I port inputs a byte from the specified I/O port address, and displays its value.

L filename[addr] loads the specified file into memory, starting at the specified address (or 100H ifno address is specified.

M addr1,addr2,addr3[,rep] moves the block of memory between addr1 and addr2 into the area starting at addr3. If the repetition factor is given, the operation is repeated that many times (useful with some EPROM programmers).

O port,val outputs the specified value to the specified I/O port address.

P addr puts the subsequently typed ASCII data into memory starting at the specified address. The data must be terminated with an ASCII EOT character (CONTROL-D).

Q quits the MONITOR command, returns to TurboDOS.

R addr1,addr2 tests the area of RAM between the two specified addresses, and diagnoses any errors detected by the test.

S filename[addr1,addr1] saves the area of memory between the two specified addresses into the specified file. If no memory addresses are given, then the bounds from the last "L" directive are used.

T addr1,addr2 types in ASCII the area of memory between the two specified addresses. Pause with space, resume with carriage-return.

V addr1,addr2,addr3 verifies that the block of memory between addr1 and addr2 is identical to the area starting at addr3. Any discrepancies are diagnosed.

W val1,val2,...,valn scans all of memory for occurrences of the specified byte string, and displays where each occurrence is located.

Y displays the highest available address in memory (below TurboDOS and the MONITOR command code).

Example:

```
0A}MONITOR
TurboDOS Monitor -- Copyright (C) 1982, Software 2000, Inc.
* Y
AFFF
* F 100,AFFF,0
* D 100,11F
0100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
* Q
0A}
```

PAUSE Command

The PAUSE command provides a way of temporarily stopping a DO file.

```
PAUSE
```

Explanation

The PAUSE command stops processing of the DO file and waits for a [RETURN] from the keyboard to continue.

Example

```
0A}PAUSE
  Hit <CR> to continue
```

PRINT Command

The PRINT command lets you control routing of your print output.

Syntax

```
PRINT {option}...
```

Explanation

The PRINT command may include one or more of the options described below. A PRINT command with no option arguments causes your current print routing to be displayed on the console. In a multi-user system, each user controls his own print routing independently.

Options

Option	Explanation
CONSOLE C	Print output is routed to your console (not to a printer)
DRIVE=d D=d	Drive "d" is used for spooling, where "d" is a drive letter in the range A through P.
FILE F	Print output is spooled to disk, but not automatically queued for printing.
OFF O	Print output is discarded.
PRINTER=p P=p	Print output is routed direct to printer "p" without spooling to disk, where "p" is a printer letter in the range A through P.
QUEUE=q Q=q	Print output is spooled to disk, and then automatically queued on queue "q" for printing, where "q" is a queue letter in the range A through P.

Examples

```
0A}PRINT DRIVE=C QUEUE=A
Printing is to SPOOLER on DRIVE C
            to QUEUE A
0A}PRINT P=B
Printing is to PRINTER B
0A}PRINT C
Printing is to CONSOLE
0A}PRINT O
Printing is to OFFLINE
0A}
```

Error Messages

```
Invalid request
```

PRINTER Command

The PRINTER command lets you control de-spooling on any selected printer.

Syntax

```
PRINTER p {option}...
```

Explanation

The "p" argument is a printer letter in the range A through P, and identifies the printer to be controlled. The PRINTER command may include one or more of the options described below. A PRINTER command with no option arguments causes the current status of printer "p" to be displayed on the console. In a multi-user system, any printer may be controlled from any console.

Options

Option	Explanation
BEGIN B	Stop de-spooling to printer, and reposition current print job to start at the beginning again when de-spooling is resumed.
GO G	Resume de-spooling to printer after STOP or BEGIN.
OFFLINE O	Take printer offline at the end of current print job. No further de-spooled printing will be done, but printer is available for direct printing.
QUEUE=q Q=q	De-spool to printer from print queue "q", where "q" is a queue letter in the range A through P. If printer is currently printing from another queue, the new assignment takes effect at the end of the current print job.

Options (Continued)

Option	Explanation
STOP S	Stop de-spooling to printer, pending subsequent GO or TERMINATE.
TERMINATE T	Terminate the current print job, and continue with the next job in the queue. The terminated print file is not deleted, and may be manually requeued with the QUEUE command.

Examples

```
0A}PRINTER B QUEUE=A
Printer B Assigned to QUEUE A
0A}PRINTER B STOP
Printer B Assigned to QUEUE A (Stopped)
0A}
```

Error Messages

```
Invalid request
```

QUEUE Command

The QUEUE command lets you manually queue print files (or any text file) for de-spooled printing.

Syntax

```
QUEUE filename {;options}
QUEUE {;options}
```

Explanation

The QUEUE command queues the file specified by "filename" for de-spooled printing.

You may use wild-card characters (? and *) in the "filename" argument to indicate that multiple files are to be queued.

If you omit "filename" from the command (second form), then the QUEUE command operates in an interactive mode. You are prompted by an asterisk * to enter a series of directives from the console. The syntax of each directive is:

```
filename {;options}
```

Options specified within a directive take precedence over options specified in the QUEUE command itself. A null directive (RETURN only) terminates the QUEUE command.

Options

Option	Explanation
;D	Queued file(s) are to be deleted automatically after printing.
;N	If "filename" contains wild cards, QUEUE does not ask you for confirmation before each file is queued.
;Q=q	File(s) are queued on print queue "q", where "q" is a queue letter in the range A through P. (If this option is omitted, the queue specified in the last PRINT command is used.)
;S	Queued file(s) are to be saved after printing, not deleted. (If neither ;D nor ;S are specified, then ;S is assumed.)
;Y	If "filename" contains wild cards, QUEUE asks you for confirmation before each file is queued.

Example

```
0A}QUEUE -PRINT-.* ;NDQ=A
 0A:-PRINT- .008 queued
 0A:-PRINT- .014 queued
 0A:-PRINT- .020 queued
0A}
```

Error Messages

```
No queue letter available
Invalid queue letter requested
File not found
Invalid filename
Insufficient memory
```

RECEIVE Command

The RECEIVE command reads one record from a FIFO and displays it on the console.

Syntax

```
RECEIVE filename
```

Explanation

The "filename" argument specifies the FIFO to be read. The RECEIVE command is useful where FIFOs are used as message mailboxes in a multi-user system.

Note that RECEIVE displays only one record. To display or print the entire contents of a FIFO, use the TYPE command instead.

Example

```
0A}RECEIVE MAILBOX.RWR
RONNIE: NEED TO MEET WITH YOU...TEDDY
0A}
```

Error Messages

```
No FIFO file name specified
FIFO file not found
File not FIFO
FIFO file empty
```

RENAME Command

The RENAME command lets you rename individual disk files or groups of files.

Syntax

```
RENAME oldname newname {;options}
RENAME {;options}
```

Explanation

The RENAME command changes the name of the file specified by "oldname" to the new name specified by "newname".

You may use wild-card characters (? and *) in the "oldname" argument to indicate that multiple files are to be renamed. Wild-card characters may also be used in the "newname" argument to indicate that the corresponding characters of the old filename are to be used in the new filename.

If you omit both "oldname" and "newname" from the command (second form), then the RENAME command operates in an interactive mode. You are prompted by an asterisk * to enter a series of directives from the console. The syntax of each directive is:

```
oldname newname {;options}
```

Options specified within a directive take precedence over options specified in the RENAME command itself. A null directive (RETURN only) terminates the RENAME command.

Options

Option	Explanation
;N	If "oldname" contains wild cards, RENAME does not ask you for confirmation before renaming each file.
;Y	If "oldname" contains wild cards, RENAME asks you for confirmation before renaming each file.

Examples

```
0A}RENAME B:*.BAK *.PRV ;N
 0B:REFERENC.BAK renamed 0B:REFERENC.PRV
 0B:USERGUID.BAK renamed 0B:USERGUID.PRV
0A}RENAME
* LETTER.TXT *.BAK
 0A:LETTER  .TXT renamed 0A:LETTER.  BAK
* MAXIMUMS.COM OPT*.*
 0A:MAXIMUMS.COM renamed 0A:OPTIMUMS.COM
* [RETURN]
0A}
```

Error Messages

```
Invalid drive letter(s)
Destination drive read only
<filename> rename to same filename
<filename> read only
File not found
Invalid filename 1
Invalid filename 2
Insufficient memory
Network error
```

SEND Command

The SEND command lets you write a message to a FIFO.

Syntax

```
SEND filename message
```

Explanation

The "filename" argument specifies the FIFO to be read, and "message" is any text you wish. The SEND command is useful where FIFOs are used as batch queues or message mailboxes in a multi-user system.

Any occurrence of the vertical bar | in the "message" argument is replaced by the character backslant \, which is helpful if you are sending a command string to a batch queue.

Examples

```
0A}SEND BATCH.DO DELETE *.BAK :N
Message sent to FIFO
0A}SEND MAILBOX.EMK TED: NO DICE...RON
Message sent to FIFO
0A}
```

Error Messages

```
No FIFO file name specified
FIFO file not found
File not FIFO
FIFO file full
```

Patch Points

Z80	8086	Val	Description
103H	CS+4	"|"	Special delimiter
104H	CS+5	"\"	Normal delimiter

SERVER Command

The SERVER command lets you temporarily attach your console to the "server" processor in a network system.

Syntax

```
SERVER
```

Note: The Turbo-Plus SERVER command will normally be used for North Star TurboDOS 8-bit systems. See the Turbo-Plus User's Guide for a description of the Turbo-Plus SERVER command.

Explanation

In many TurboDOS systems, disks and other shared peripherals are attached to a "server" processor which is separate from the "user" processors to which user consoles are attached. If you need to run a program in the server processor (the BACKUP, BOOT, FORMAT or VERIFY commands, for example), you can use the SERVER command to assign your console to the server temporarily. To detach from the server (and resume normal console operation), enter a [BREAK] [CONTROL-C] sequence.

While attached to the server, you can send attention requests to the server processor by using a special attention character (CTRL-A) instead of the regular one (CTRL-S).

Use of the SERVER command is restricted to privileged log-ons only. Do not attempt to run SERVER from more than one console at a time. If you do, console output from the server will be randomly distributed across two or more consoles, and be undecipherable. If this should occur by mistake, simply detach all but one of the consoles.

Example

```
5C}SERVER
Console attached to server processor
0A}FORMAT B:
        :
        :
0A}[CTRL-S][CTRL-C]
Console detached from server processor
5C}
```

Error Messages

```
Console already attached to server
Non-privileged user
Remote console driver not present
```

Patch Points

Z80	8086	Val	Description
103H	CS+4	"^A"	Special attention char
104H	CS+5	"^S"	Normal attention char

SET Command

The SET command lets you set and clear file attributes.

Syntax

```
SET filename ;options
SET {;options}
```

Explanation

The SET command changes the attributes of the file specified by the "filename" argument as requested in "options".

You may use wild-card characters (? and *) in the "filename" argument to indicate that attributes of multiple files are to changed.

SET may also be used to set a drive read-only or read/write by specifying a "filename" consisting of a drive letter "d:" only.

If you omit "filename" from the command (second form), then the SET command operates in an interactive mode. You are prompted by an asterisk * to enter a series of directives from the console. The syntax of each directive is:

```
filename {;options}
```

Options specified within a directive take precedence over options specified in the SET command itself. A null directive (RETURN only) terminates the SET command.

Options

Option	Explanation
;+A	Set "archived" attribute
;+F	Set "FIFO" attribute
;+G	Set "global" attribute
;+R	Set "read-only" attribute
;-A	Clear "archived" attribute
;-F	Clear "FIFO" attribute
;-G	Clear "global" attribute
;-R	Clear "read-only" attribute
;N	If "filename" contains wild cards, SET does not ask you for confirmation before changing attributes of each file.
;Y	If "filename" contains wild cards, SET asks you for confirmation before changing attributes of each file.

Examples

```
0A}SET *.COM ;N +RG -A
 0A:COPY    .COM set
 0A:DELETE  .COM set
 0A:RENAME  .COM set
0A}SET B: ;+R
Drive B set to read-only
0A}SET B: ;-R
Drive B set to read/write
0A}
```

Error Messages

```
Destination drive read only
File not found
Invalid filename
Invalid attribute
Insufficient memory
Network error
```

SHOW Command

The SHOW command lets you display the settings of file attributes.

Syntax

```
SHOW filename ;options
SHOW {;options}
```

Explanation

The SHOW command displays the attributes of the file specified by the "filename" argument.

You may use wild-card characters (? and *) in the "filename" argument to indicate that attributes of multiple files are to be displayed.

SHOW may also be used to display the status of a drive (read-only or read/write) by specifying a "filename" consisting of a drive letter "d:" only.

If you omit "filename" from the command (second form), then the SHOW command operates in an interactive mode. You are prompted by an asterisk * to enter a series of directives from the console. The syntax of each directive is:

```
filename {;options}
```

Options specified within a directive take precedence over options specified in the SHOW command itself. A null directive (RETURN only) terminates the SHOW command.

Options

Option	Explanation
;L	SHOW prints the file attributes (rather than displaying them on the console).
;N	If "filename" contains wild cards, SHOW does not ask you for confirmation before displaying attributes of each file.
;Y	If "filename" contains wild cards, SHOW asks you for confirmation before displaying the attributes of each file.

Examples

```
0A}SHOW *.COM ;N
 0A:COPY    .COM has attributes: AGR
 0A:DELETE  .COM has attributes: AG
 0A:RENAME  .COM has attributes: A
0A}SHOW B:
Drive B is read-only
0A}SHOW A:
Drive A is read/write
0A}
```

Error Messages

```
File not found
Invalid filename
Insufficient memory
```

TYPE Command

The TYPE command displays the contents of a text file on the console or printer.

Syntax

```
TYPE filename {;options}
```

Explanation

The TYPE command displays the contents of the text file specified by "filename". The result may be directed to either the console or printer.

Options

Option	Explanation
;L	The file is printed, rather than displayed on the console.

Example

```
0A}TYPE B:UPDATE.DOC
...(display of B:UPDATE.DOC)...
0A}
```

Error Messages

```
File not found
```

USER Command

The USER command lets you change the current user number.

Syntax

```
USER {nn}
```

Explanation

The "nn" argument specifies the desired user number (in the range 0 to 31). If "nn" is omitted, the command displays the current user number (which also appears in the command prompt).

The USER command is restricted to privileged log-ons only.

Examples

```
0A}USER 2
Current user number: 2
2A}USER 0
Current user number: 0
0A}
```

Error Messages

```
Invalid user number requested
Non-privileged user
```

VERIFY Command

The VERIFY command scans a disk for bad blocks, and (optionally) marks them so that TurboDOS will avoid using them.

Syntax

```
VERIFY d: {;options}
```

Explanation

The VERIFY command reads every block on the disk identified by "d:", which must be attached to the processor in which the command is executed. If bad blocks are detected, VERIFY will tell you how many of them were encountered, and will ask you whether you want them marked. If you answer "yes", then VERIFY will create a read-only directory entry under the name BLOCKS.BAD to reserve the defective blocks and prevent their subsequent use by TurboDOS. Use of the VERIFY command is restricted to privileged log-ons only.

Options

Option	Explanation
;R	VERIFY repeats continuously (allowing multiple disks to be verified) until terminated by typing CTRL-C.

Example

```
0A}VERIFY N:
Disk to be verified is in drive N
Enter <CR> to begin verifying: [RETURN]
Verifying blocks on drive: N
.....
Verify completed - 3 bad blocks found
Mark bad blocks in directory (y/n)? Y
Marking bad blocks in directory
0A}
```

Error Messages

```
Non-privileged user
Unable to execute from bank 1
No verify drive specified
Unable to lockout verify drive
Verify drive not ready
Insufficient memory to verify
Excessive bad blocks
Error marking bad blocks
Bad directory block
```

YES Command

The YES command causes the word 'YES' to be displayed on the screen.

Syntax

```
YES
```

Explanation

The YES command is used in conjunction with VERIFY in a DO file. When using VERIFY to generate Bad Blocks file:

- A 'yes' is needed to generate the file if any bad blocks are found. A YES command following the VERIFY serves this purpose in a DO file.
- For any other case, including no bad blocks found, YES prints on the display but has no other action.

8086 family: A family of compatible 16-bit microprocessors designed by Intel Corporation, which include the 8086, 8088, 80186 and 80286.

abort: Premature termination of a program at your request. TurboDOS provides a special attention sequence which enables you to abort a program at any time.

allocation map: The area on a disk where TurboDOS keeps track of which portions of the disk storage space are occupied with information, and which portions are free.

application program: Computer program designed for a particular application, such as word processing, data management, accounting, spreadsheet calculations, or statistical analysis.

archived attribute: An file attribute which indicates that a file has not been changed since the last time it was backed up, and therefore does not have to be backed up again.

argument: Variable item in a command which is passed to the command program. Arguments may specify a drive, a file name, a command option, etc.

ASCII: American Standard Code for Information Exchange is a code used to represent textual information in computers. ASCII defines a standard representation for each letter, digit, and punctuation character as a 7-bit binary number.

attention request: A means for interrupting the execution of the program in progress in order to make a request of TurboDOS. In most systems, you issue an attention request by keying either CTRL-S or BREAK.

attribute: A file characteristic that you can set or clear. TurboDOS supports four attributes: archived, read-only, global, and FIFO.

autoload: A facility which allows a program or command sequence to be executed automatically without explicit user action.

automatic spooling: A printing method in which print output is first accumulated in a disk file, then automatically queued for background printing.

backspace: A key on most console keyboards used to correct typing errors by deleting incorrect keystrokes.

backup: Copy of a file or disk made to protect against loss in case of human error or equipment failure. Also, the act of making such backup copies.

bad block: Spot on a disk which is defective and unusable. TurboDOS allows such bad blocks to be detected, marked, and avoided.

banked memory: A hardware technique in which multiple memory banks are installed in a microcomputer, together with a mechanism for switching from one bank to another under software control. This approach is commonly used to allow more than 64K of memory to be used with an 8-bit microprocessor such as the Z80.

batch processing: Processing under the control of a pre-defined sequence of commands (see "do-file").

bit: Short for "binary digit" which can take the values 0 or 1, the smallest unit of information. (See "byte", "kilobyte", and "megabyte".)

boot tracks: See "reserved tracks".

buffer: An area of memory used to hold data temporarily during processing. TurboDOS provides an elaborate disk buffer facility in order to speed performance by reducing the frequency of disk accesses.

byte: Basic unit of information, consisting of 8 bits. A byte of information can represent one ASCII character, or an integer between 0 and 255. (See "bit", "kilobyte", and "megabyte".)

cold start: Initial start-up of TurboDOS, which involves reading a copy of the TurboDOS operating system from disk into the computer's memory.

command: An instruction you give to TurboDOS, telling it to run a particular program. Also, any of the more than 30 standard programs furnished with the TurboDOS operating system.

command prompt: A symbol (e.g., "0A}") displayed on your console to indicate that TurboDOS is ready to accept a command. The command prompt consists of the current user number (e.g., "0"), the current drive letter (e.g., "A"), and the character "}".

command string: A sequence of TurboDOS commands that you key in together on a single line. The commands in a command string must be separated by the backslant \ character.

command tail: Arguments which follow the program name in a command. TurboDOS passes the command tail to the program.

confirmation: Feature of various TurboDOS commands that allows you to approve or cancel each file operation before it is done.

console: Hardware device that allows you to interact with your computer, usually consisting of a keyboard and a display screen.

CP/M: Short for "Control Program/Microcomputers", a widely-used operating system for single-user microcomputer systems. CP/M is a registered trademark of Digital Research Inc.

CTRL: A special key on most console keyboards for entering non-printing control characters. To enter the character "CTRL-X", you have to hold down the CTRL key while you press X.

current drive: The disk drive which is used by default whenever you don't specify any drive explicitly.

cursor: An indicator on your console screen (usually a box or underline) which shows you where the next character you type will appear.

data: Any kind of information (numbers, text, formulas, pictures, etc.) that may be used as input to or output from a computer program.

default: The value assumed by TurboDOS implicitly when you do not specify an argument explicitly.

delimiter: Character that separates one argument or command from another.

de-spooling: Background printing from a print file on disk.

diagnostic: A message displayed on your console to tell you that a processing error has occurred, and often prompting you to choose among alternative methods of recovering from the error.

direct printing: Routing of print output directly to a printer on a character-by-character basis. (See "spooling".)

directory: A table of contents maintained by TurboDOS on each disk to keep track of files stored on the disk.

disk: A rotating magnetic medium used by computer systems to store programs and data. (See "floppy disk" and "hard disk".)

diskette: See "floppy disk".

do-file: A file containing a pre-defined sequence of TurboDOS commands. (See "batch processing").

drive: A hardware device which reads and writes on magnetic disks, (much as a tape recorder records and plays magnetic tape). TurboDOS identifies each drive with a letter between A and P.

dump: A display or printout of information consisting of the hexadecimal representation of each byte of data. (See "hexadecimal").

echo-print: A mode in which every character displayed on the console is also echoed to the printer.

FIFO: Short for "first-in, first-out". A special kind of file in which data is always written to the end but read from the beginning. FIFOs are used mostly for inter-processor and inter-user communications.

FIFO attribute: File attribute which designates that a file is to be accessed using the special FIFO access method.

file: Collection of related information stored on disk. Files may contain programs or data.

file attribute: See "attribute".

file lock: Mechanism to prevent several users from trying to access or modify a file at the same time.

file name: The name you assign to a file so that you can identify it later. TurboDOS keeps track of your files by name.

file type: An optional extension to a file name used to group similar files together.

floppy disk: Inexpensive removable disks which use a flexible magnetic medium that spins inside a cardboard jacket. Floppy disks come in 8-inch and 5.25-inch diameter sizes (smaller ones are being introduced). They may be recorded on one or both sides, and commonly have capacities of 150 to 1,500 kilobytes.

format: The way in which data on disk is organized into tracks and sectors. Also, the process of initializing a disk with the desired format.

global attribute: File attribute which designates that a file saved under user number 0 is to be accessible from any user number. Global files provide a convenient way to make selected files available to all users. For example, TurboDOS commands and other common programs are often given the global attribute.

hard disk: High-performance disks which use one or several rigid magnetic platters which spin inside a sealed chamber. Hard disks offer much greater storage capacity than floppy disks, typically 5 to 50 megabytes or more.

hard sectored: Disk whose format is fixed by the hardware itself.

hardware: Physical equipment comprising a computer system (as distinguished from the programs that run on it).

hashed directory: A directory format that uses a sophisticated technique to make directory look-up much faster than possible with a linear directory. Hashed directories are especially suited for use on hard disks with big directories.

hexadecimal: Numeric notation using base 16, which is often a convenient way to represent non-textual information. The sixteen hexadecimal digits are represented by the characters 0-9 and A-F.

incremental backup: An efficient backup procedure whereby backup copies are made only of files which have been created or modified since the last backup cycle. (See "archived attribute".)

kilobyte: Unit of information consisting of 1,024 bytes and abbreviated by the letter "K". Each page of this document contains about 3K of information. (See "byte".)

linear directory: A simple directory format compatible with CP/M and suitable for use on floppy disks. (See "hashed directory".)

local command: A command which requires direct access to disk drive and controller hardware, and consequently may be executed only in the processor to which the disk is attached.

log-off: Procedure for terminating a console session on a multi-user system. After log-off, a log-on is required before your console can be used again. (See "log-on".)

log-on: Procedure for starting a console session on a multi-user system. You must identify yourself by entering your user-id and possibly a security password before TurboDOS will allow you to make use of the computer.

loosely-coupled network: Multi-processor system in which the various processors are physically separated (often built into user consoles), and connected with some sort of communications path.

manual queuing: Submitting files for background printing by using the QUEUE command.

manual spooling: A printing method in which print output is accumulated in a disk file, but not automatically queued for background printing.

megabyte: Unit of information consisting of 1,024 kilobytes (1,048,576 bytes). The entire set of TurboDOS manuals contains roughly a megabyte of information. (See "byte" and "kilobyte".)

memory: Hardware in a computer system which stores information. (See "random-access memory".)

MP/M: Multi-user version of the CP/M operating system, which uses time-sharing to support multiple users. MP/M is a registered trademark of Digital Research Inc. (See "CP/M", "time-sharing".)

multi-processor: System composed of several microcomputers networked together. (See "microprocessor", "networking".)

multi-user: System capable of supporting several user activities at several consoles simultaneously. (See "networking", "time-sharing".)

nested do-file: A do-file executed from within another do-file.

networking: Coordinating a system of interconnected microcomputers. Generally, a separate microcomputer supports each user console. One or more additional microcomputers manage the disks, printers, and other shared devices of the system.

operating system: A program which supervises the operation of application programs, and which manages memory, disk drives, consoles, printers, and other hardware resources of a computer system. TurboDOS is an operating system.

parameter: A place-marker in a do-file that indicates where a variable argument is to be substituted when the do-file is executed. (See "argument", "do-file".)

password: A security code used to validate a user-id during the log-on procedure. (See "log-on", "user-id".)

peripheral device: A hardware device attached to a microcomputer for purposes of input or output of information. Typical peripheral devices include disk drives, consoles and printers.

print job: A unit of printed output, generally the output produced by one application program.

print queue: A list of print jobs to be printed in the background.

printer: Hardware device that produces printed hardcopy of textual information.

privileged log-on: A user who has logged-on successfully to a privileged user-id, enabling him to access various protected commands and features of TurboDOS.

program: Instructions for performing a processing task, coded so that they can be interpreted by the computer hardware.

prompt: Symbol displayed on the console by a program to advise you what to do next.

queue: See "print queue".

random-access memory (RAM): The primary high-speed memory of a microcomputer, used to hold programs in execution and data being processed.

read-after-write: When writing data to disk, immediately reading it back to make sure it was recorded properly.

read-only attribute: File attribute which designates that a file may be read but not modified or deleted. This helps you protect important files against accidental distruction. A drive may also be set read-only.

record: Grouping of related data within a file. In TurboDOS, often refers to a block of data 128-bytes long.

record lock: Mechanism to prevent several users from trying to access or modify the same record of a file at the same time.

recovery: Procedure to restart processing after encountering an error.

remote console: Using a console attached to a slave processor to control activities in a master processor.

reserved tracks: Designated tracks on a disk required by certain hardware configurations to support system start-up, and not otherwise used by TurboDOS.

sector: Portion of a track on disk. A disk is formatted into a fixed number of sectors per track.

server processor: In a networking system, a processor whose primary function is to manage disk drives, printers, and othe shared peripherals on behalf of other (user) processors.

soft sectored: Disk whose format is determined by pre-recording formatting information onto the disk.

software: Computer programs (as distinguished from the hardware they run on).

spooling: Routing of print output to a disk file, instead of directly to a printer. (See "de-spooling", "direct printing".)

syntax: Format for entering a given command.

tightly-coupled network: Multi-processor system in which the various processors are co-located. (See "loosely-coupled network".)

time-sharing: Technique for supporting multiple users with one microprocessor, whereby the processing time of the single processor is shared among the various users.

track: Concentric ring on a disk, upon which data is magnetically recorded. A track is subdivided into sectors.

transient program area (TPA): Memory space available to run commands and application programs.

user-id: Identification of an authorized user, used for security purposes on a multi-user system.

user number: Number from 0 to 31 assigned to a file which it is created. User numbers can be used to organize files into separate libraries.

user processor: In a networking system, a processor whose primary function is to interact with a console and run applicatiion programs. (See "server processor".)

volume label: The name you assign to a particular disk.

warm start: Termination of a command or application program.

wild card: Special characters (? and *) used in a file name to match any character in the corresponding position.

Winchester disk: A non-removable form of hard disk, in which rigid magnetic platters spin inside of a hermetically sealed vessel to ensure freedom from contamination.

word processor: An application program designed for composing, editing, printing, and otherwise manipulating text.

Z80: An 8-bit microprocessor created by Zilog, Inc.