Software 2000
oftware 2000
ftware 2000
tware 2000
ware 2000
are 2000
re 2000
e 2000
 2000
2000
000
00
0

User's Guide to TurboDOS

T
Tu
Tur
Turb
Turbo
TurboD
TurboDO
TurboDOS

User's Guide to TurboDOS

September, 1981

TABLE OF CONTENTS

OVERVIEW

This User's Guide to TurboDOS provides the information that users need to write and run programs under the TurboDOS operating system. It includes an overview of operating system features, a discussion of architecture and theory of operation, a description of each command, and a definition of each user-callable function.

A companion document, entitled Configuration Guide to TurboDOS, provides the information that OEMs and sophisticated users need to generate various operating system configurations and to implement driver modules for various peripheral components.

Principal Features

TurboDOS is a state-of-the-art operating system designed for use as a superior alternative to Digital Research CP/M, MP/M and CP/NET on Z80-based microcomputers. Major features of TurboDOS are outlined below:

CP/M Compatibility. TurboDOS is compatible with Digital Research CP/M Version 2.2, and can be used as a direct replacement for CP/M on any Z80-based microcomputer. Any CP/M application, language processor or programming tool should operate under TurboDOS without modification. CP/M BDOS functions and direct BIOS calls are fully supported. TurboDOS is also fully media-compatible with CP/M. It automatically determines whether a diskette is written in standard CP/M format or in TurboDOS format. (TurboDOS format diskettes hold 25% to 35% more data, and run very much faster.) The format of a newly-created diskette is determined when the diskette is initialized.

Modular Architecture. TurboDOS consists of a set of "building block" modules which may be combined to produce a family of compatible operating systems, including single-task, spooling, multi-task and networking. Each functional area of the operating system is packaged a separate relocatable module, as is each hardware-dependent device driver. Modular architecture makes it much easier to adapt TurboDOS to various hardware configurations. OEMs who wish to revise any portion of the operating system may purchase TurboDOS source code on a module-by-module basis.

Improved Performance. In comparative performance benchmarks, TurboDOS is much faster than CP/M in file-oriented applications. Much of this speed advantage is accomplished by a sophisticated buffer manager. This module performs multi-level buffering of disk I/O, using least-recently-used (LRU) buffer assignment and other sophisticated optimizations. Buffering provides a manyfold reduction in the number of physical disk accesses in both sequential and random operations. Additional speed is provided by a program load optimizer which scans the allocation map of a program file, determines the sequentially allocated segments of the file (often 16K or more in length), and loads these segments at the maximum transfer rate of the disk controller. Another major performance improvement is the elimination of warm-start and disk log-on delays. Warm-start is instantaneous in TurboDOS because the command interpreter is resident (not transient as in CP/M). Disk log-on is not required in TurboDOS because the allocation map for each disk is stored on the disk (and need not be repeatedly recreated as in CP/M). Written specifically for the Z80, TurboDOS takes full advantage of the extended instruction repertoire of the Z80 to increase speed and reduce memory overhead. Use of Z80 index registers has made it practical for the TurboDOS kernel to be fully re-entrant, providing greatly improved performance in spooling and multi-user operations.

Increased Disk Capacity. TurboDOS lets you store 25% to 35% more data on each floppy disk, compared to most CP/M implementations. Most of the increased diskette capacity is achieved through the use of larger physical sector sizes on diskette. Additional capacity is achieved by eliminating reserved "system tracks". To provide compatability, standard-format CP/M diskettes are also accomodated by TurboDOS. TurboDOS was designed to support large hard disks. It supports hard disk drives in excess of 1,000 megabytes without partitioning, and allows random access to files up to 134 megabytes.

Enhanced Reliability. TurboDOS performs read-after-write verification of all disk update operations. While this kind of verification has long been standard practice on large-scale computer systems, it is seldom seen on microcomputer systems. The TurboDOS buffer manager makes it possible to perform read-after-write verification without degrading performance to an intolerable degree. Whenever errors are detected, TurboDOS provides meaningful diagnostic messages and a variety of recovery options. In the event of a disk error, for example, TurboDOS diagnoses the drive, track and sector involved, and prompts the operator to choose one of three recovery alternatives: (1) retry the disk operation again, (2) accept the error and continue processing, or (3) abort the program. TurboDOS does not depend on the availability of a particular disk drive, but rather scans automatically to allow system start-up from any disk drive. This feature allows normal system operation even if the "A" drive fails.

Simplified Disk Changes. The allocation map for each disk is maintained by TurboDOS on the disk itself. Consequently, you can change any disk at any time without fear of the disk becoming "read-only" or the data being compromised.

TurboDOS senses and automatically adapts to changes of disk format (one- or two-sided, single- or double-density, etc.) Hassle-free disk changes under TurboDOS make low-cost single-disk systems truly practical.

Print Spooling. TurboDOS includes high-performance automatic print spooling as an integral part of the operating system. Up to 16 concurrent printers and up to 16 separate print queues are supported.

Additional Capabilities. The command interpreter of TurboDOS accepts strings of multiple commands, not just single commands. Its command file processor is much more powerful than the "submit" facility of CP/M, and allows multi-level nesting of command files. TurboDOS provides an extensive set of utility programs, system date and time functions, standard communications channel interface, and numerous other capabilities not available in CP/M.

Multi-User Facilities. Multi-user configurations of TurboDOS provide the file and record interlocks necessary to permit simultaneous multi-user access to common data bases. Implicit file-level interlocks are completely automatic, require no user-program participation, and support multiple inquiry tasks concurrent with one update task. Explicit record-level interlocks support concurrent interleaved update by multiple tasks. Password-type log-on security prevents unauthorized access and protects private file libraries. A log file keeps an automatic record of all system usage.

Networking Facilities. TurboDOS supports a multi-user network of interconnected microcomputers which can share a common pool of mass storage, printers and other peripherals. Since this approach provides a microcomputer dedicated to each user, TurboDOS is able to support a large number of simultaneous users with excellent performance. TurboDOS is especially well-suited to the new generation of bus-oriented multi-processor microcomputer hardware which is now becoming available. Both master-to-master and master-to-slave interconnections are supported. Slave processors may have their own local disk storage, or may be entirely down-line loaded from a master processor. Networking TurboDOS incorporates an advanced failure detection and recovery facility that makes a master-slave network virtually crashproof. Even a malicious user running in a slave processor cannot compromise the processing or files belonging to another user.

Operational Differences from CP/M

Operation under TurboDOS is similar to CP/M, with the following significant differences:

1. Program loading and all file operations are very much faster under TurboDOS. Warm-start (when a program returns to the operating system) and disk log-on (when a drive is accessed for the first time after a warm-start) are instantaneous in TurboDOS. As a result, you will notice a different operating "rhythm".

2. The command prompt is of the form "0A}" (instead of CP/M's "A>") to provide a constant reminder that you are using TurboDOS. The number indicates the current user number, and the letter indicates the default drive.

3. In response to a command prompt, TurboDOS allows you to enter either single commands or multi-command strings. Multiple commands must be separated by a command separator character, normally backslant "\". The length of a multi-command string is limited to the size of the command buffer, normally 157 characters. If you enter a multi-command string, TurboDOS displays each command in the string as it is executed.

4. All TurboDOS commands are transient programs (i.e., .COM files). There are no "built-in" commands. Each command is described in detail in Section 3 of this document. Many of the TurboDOS commands are rather different than the ones supported by CP/M.

5. The TurboDOS "DO" command enables you to execute a previously prepared file of commands. This facility works somewhat differently than CP/M's "SUBMIT", and permits DO-file nesting to any reasonable depth. See the description of the "DO" command for details.

6. Under TurboDOS, you can stop the execution of a program or DO-file by typing the attention character (normally CTRL-S). TurboDOS will "beep" in response to such an attention request. After an attention request, you may restart execution by typing the restart character (normally CTRL-Q), or abort by typing the abort character (normally CTRL-C).

7. TurboDOS provides a flexible and user-defined facility for loading any program (e.g., a function menu) automatically at initial system cold-start and/or at each system warm-start. See the description of the "AUTOLOAD" command for details.

8. Under single-user TurboDOS, you may safely change disks any time the system is awaiting input from the console keyboard. A disk will never become "read-only" as the result of a disk change. It is not necessary to type CTRL-C when changing disks; in fact, TurboDOS will ignore CTRL-C in most contexts. When processing under the control of a DO-file, you should wait for the DO-file to complete before changing disks. When doing concurrent printing, wait for all printing to complete before changing the disk in the de-spool drive. Under multi-user configurations of TurboDOS, you should enter a "CHANGE" command before changing disks, to ensure that no other user is affected by the disk change.

9. TurboDOS-format diskettes have large sector sizes (512 or 1024 bytes), no skew, and no reserved tracks. TurboDOS can also use standard CP/M-format diskettes; however, performance and disk capacity are greatly reduced. Therefore, it is a good idea to copy programs and data to TurboDOS-format diskettes whenever possible. The format of a diskette is determined when the diskette is initialized, and is sensed automatically and dynamically by TurboDOS after any disk change.

Start-Up Procedure

If your computer has a TurboDOS start-up PROM installed, system start-up is accomplished by resetting the processor and inserting a TurboDOS-format disk which contains the files OSLOAD.COM and OSMASTER.SYS in any drive. The start-up process is automatic. When start-up is complete, a sign-on message is displayed.

If your computer has a CP/M start-up PROM installed, system start-up is a two-stage process. First, you must bring up CP/M in the usual fashion. Then, you must insert a CP/M-format disk which contains the files OSLOAD.COM and OSMASTER.SYS in any drive, log-on to that drive, and type "OSLOAD".

THEORY OF OPERATION

Modular Architecture

TurboDOS is packaged as a set of relocatable modules. Each functional area of the operating system is packaged a separate module, as is each hardware-dependent device driver. As an illustration, some of the functional modules of TurboDOS are:

- Local user interface
- Slave network interface
- Master network interface
- Command language interpreter
- Command file processor
- OS function decode
- Non-file manager
- File manager
- Program load optimizer
- Memory manager
- Buffer manager
- Disk manager
- Console manager
- Printer manager
- Multi-task dispatcher
- Spooler/despooler
- Physical console manager
- Physical printer manager
- Physical floppy disk manager
- Physical hard-disk manager

Many of these modules are optional. The modules are "building blocks" which can be combined in various ways to produce a family of compatible operating systems. Possible TurboDOS configurations include single-task, spooling, multi-task, real-time, time-sharing, distributed-processing, and networking.

Modular architecture facilitates the adaptation of TurboDOS to various hardware configurations. Each hardware-dependent element is a separate relocatable module. An easy-to-use system generation utility program is used to link together the desired combination of functional and hardware-dependent modules to create a version of TurboDOS configured with the desired features and for the desired hardware.

CP/M-Compatible Functions

TurboDOS functions are invoked by a call to location 5 with a function number in the C-register, in the same fashion as in CP/M. TurboDOS supports the following CP/M functions:

0	Return to O/S	20	Read Next Record
1	Input Character from Console	21	Write Next Record
2	Output Character to Console	22	Create File
3	Input Character from Reader	23	Rename File
4	Output Character to Punch	24*	Return Ready Vector
5	Output Character to Printer	25	Return Default Disk
6	Direct Console I/O	26	Set Record Buffer Address
7	Return I/O Routing	27*	Return Disk Allocation Info
8	Set I/O Routing	28	Set Write-Protect Vector
9	Output Buffer to Console	29	Return Write-Protect Vector
10	Input Buffer from Console	30	Set File Attributes
11	Return Console Status	31*	Return Physical Disk Info
12	Return O/S Version	32	Set/Return User Number
13	Reset Disk System	33	Read Random Record
14	Select Default Disk	34	Write Random Record
15	Open File	35	Compute File Size
16	Close File	36	Set Random Record
17	Search for First File	37	Reset Write Protect Vector
18	Search for Next File		(38,39 are Reserved)
19	Delete File	40*	Write Random w/Zero Fill

These TurboDOS functions are the exact equivalents of the corresponding functions in CP/M, except for the four functions marked with an asterisk above. Three of these functions (24, 27 and 31) affect only the CP/M "STAT" utility, which is not normally used with TurboDOS. Refer to Section 4 for details of the TurboDOS implementation of these functions. Function 40 is treated by TurboDOS as a synonym for function 34.

TurboDOS also provides a full simulated "BIOS branch table" in order to support programs which make direct calls on the CP/M BIOS. This branch table always begins 256 bytes from the top of memory in order to ensure that it begins on a page boundary (as it does in CP/M).

Additional Functions

In addition to the CP/M-compatible functions listed above, TurboDOS supports more than 40 additional functions. These are numbered in the range 64-127 to avoid conflict with CP/M and MP/M. They are concerned with real-time clock, communications channel support, buffer management, DO-files, autoload, spooling, disk labels, networking, log-on/log-off and other TurboDOS extensions. These additional functions are provided primarily to support various TurboDOS commands, but they may be invoked by user programs if desired. Refer to Section 5 for a detailed description of each function.

Disk Formats

TurboDOS was designed to accomodate any combination of disk storage units, from mini-floppies to large hard disks in excess of 1,000 megabytes. For maximum capacity and performance, TurboDOS disks are generally formatted with large sector sizes (typically 512 or 1,024 bytes), no skew, and no reserved "system tracks". However, TurboDOS is also fully media-compatible with CP/M. It automatically determines whether a diskette is written in standard CP/M format or in TurboDOS format. (TurboDOS format diskettes hold 25% to 35% more data, and run very much faster.) The format of a newly-created diskette is determined when the diskette is initialized.

The first portion of any disk is reserved for a file directory. Unlike CP/M, TurboDOS maintains a volume label and an allocation map at the beginning of the directory area. When a CP/M disk is first accessed by TurboDOS, the first few directory entries are automatically relocated 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 entries in the directory.

Command Line Processing

All TurboDOS commands are transient programs (i.e., .COM files). There are no "built-in" commands. Each command is described in detail in Section 3 of this document. Many of the TurboDOS commands are rather different than the ones supported by CP/M.

The command line interpreter module of TurboDOS accepts either single commands or multi-command strings. Multiple commands must be separated by a command separator character, normally backslant "\". The length of a multi-command string is limited to the size of the command buffer, normally 157 characters. When processing a multi-command string, TurboDOS displays each individual command on the console as it is executed.

Each TurboDOS command consists of a program file reference followed by an optional command tail up to 127 characters long. The program file reference may include an explicit file type, but ".COM" is assumed if it doesn't. The command line interpreter loads the program file into the TPA (transient program area of memory starting at 0100H) and transfers control to it. The command tail length and text are passed in the default buffer (starting at 0080H), and up to two file references in the command tail are passed in the default FCB (file control block starting at 005CH and 006CH).

A file reference identifies a particular file or group of files, and is always given in the following format:

d:filename.typ

where "d:" specifies the drive letter A...P, and may be omitted if the file is on the default drive; "filename" specifies the file name of 8 characters or less; and ".typ" specifies the file type of 3 characters or less, prefixed by a period. The character "?" may be used in the file name or file type as a "wild-card" to match any character in the corresponding position. The character "*" may be used in the file name or file type to indicate that all remaining character positions are wild-cards. Wild-cards are especially useful for making reference to a group of files without enumerating each one individually (e.g., in the COPY, DELETE and RENAME commands).

The command line interpreter is an optional module of TurboDOS, and may be omitted in dedicated application-oriented systems. Omitting this module reduces the size of the operating system, and prevents the user from escaping from the control of the application software. In absence of the command line interpreter, the TurboDOS autoload facility must be used to load the first program at system start-up.

DO-File Processing

The DO-file manager module of TurboDOS and the DO command permit automatic execution of a pre-defined sequence of TurboDOS commands stored on disk. A DO-file is simply an ASCII file of any desired length, each line of which contains any valid TurboDOS command or multi-command string. A DO-file may contain any number of embedded DO commands, even in multi-command strings. TurboDOS supports full nesting of DO-files to any reasonable depth. A DO-file may also include any number of parameters which are automatically replaced by the corresponding arguments from the tail of the DO command. DO-files can be prepared with any conventional text editor.

Certain commands (such as COPY, RENAME and DELETE) and other programs expect interactive input from the console keyboard. If such a command or program is executed within a DO-file, then its console input comes from the DO-file rather than the keyboard. An exception is console input solicited by means of direct console I/O (function 6) or direct calls on the BIOS vector.

The DO-file manager is an optional module of TurboDOS, and may be omitted in dedicated application-oriented systems. Omitting this module reduces the size of the operating system.

Attention Requests

Under TurboDOS, the execution of a program or DO-file may be suspended by typing the attention character (normally CTRL-S) at the console keyboard. TurboDOS will "beep" to acknowledge receipt of the request. After an attention request, TurboDOS will accept one of the following attention responses:

Abort (normally CTRL-C) aborts execution of the current program or DO-file, and causes any nested command lines and DO-files to be disregarded.

Echo-Print (normally CTRL-P) restarts execution, and causes all subsequent console output to be echoed to the printer (in addition to being displayed). A second Attention-Echo sequence turns off echoing of console output to the printer.

End-Print (normally CTRL-L) restarts execution after signalling the end of the current print job. If printing is being spooled, this causes the current print file to be closed and queued for printing.

Resume (normally CTRL-Q) restarts execution.

Print Spooling

The concurrent printing facilities of TurboDOS consist of a "spooler" and "de-spooler". The spooler permits print output to be redirected to disk, creating print files which are then queued for printing by the de-spooler. The de-spooler prints from its queue of print jobs concurrently with ordinary foreground processing tasks.

TurboDOS supports up to 16 simultaneous printers (A,B,C,...,P) and up to 16 print queues (A,B,C,...,P). Various print queues may be used to group together print jobs with similar forms requirements and/or priorities. Alternatively, multiple printers may be assigned to the same queue to achieve automatic load-sharing.

The PRINT command controls the routing of print output:

<u>PRINT DRIVE=d QUEUE=q</u> causes print output to be spooled to print files on the specified drive (d=A...P), then queued automatically on the specified print queue (q=A...P) for de-spooled printing. Print files are created automatically, and named -PRINT-q.001, -PRINT-q.002, etc. Print files are closed at the conclusion of each program (i.e., warm-start), by an end-print attention request from the console, by an explicit operating system function call, or by the presence of an end-of-print character in the print output stream (if EOPCHR is defined).

<u>PRINT DRIVE=d FILE</u> causes print output to be spooled to print files on the specified drive (d=A...P). The print files are <u>not</u> queued automatically for printing, but may be queued manually with the QUEUE command. Print files are created automatically, and named -PRINT-.001, -PRINT-.002, etc.

<u>PRINT PRINTER=p</u> causes print output to be routed directly to the specified printer (p=A...P) without intermediate spooling to disk.

<u>PRINT CONSOLE</u> causes print output to be routed to the console.

<u>PRINT OFF</u> causes print output to be discarded.

In a multi-user TurboDOS configuration, each user may control his own print routing independently. In a networking configuration, slave processors may route print output to a local (slave-owned) or remote (master-owned) printer by using the keywords LOCAL or REMOTE in any PRINT command.

De-Spooled Printing

The PRINTER command controls de-spooled printing:

PRINTER p QUEUE=q causes the specified printer (p=A...P) to take its print jobs from the specified de-spool queue (q=A...P). If the printer is currently printing from another queue, then the new assignment takes effect at the end of the current print job.

PRINTER p OFF causes the specified printer to be taken offline at the end of the current print job. An offline printer may be accessed for direct printing.

PRINTER p STOP temporarily suspends de-spooling to the specified printer (e.g., to correct a paper jam).

PRINTER p GO resumes de-spooling to the specified printer.

PRINTER p BEGIN stops de-spooling to the specified printer, and causes the current print job to be reprinted from the beginning when de-spooling is resumed.

PRINTER p TERMINATE terminates the current print job on the specified printer, and continues with the next queued job. The terminated print file is not deleted from disk, however, so the job may be manually requeued with the QUEUE command.

The spooler and de-spooler are optional modules of TurboDOS, and may be omitted in systems where memory space is at a premium.

Log-On and Log-Off

The LOGON and LOGOFF commands provide password-type security, the purpose of which is to prevent unauthorized access to the system and to protect private file libraries.

The LOGON command prompts for a user-ID to be entered from the console keyboard. The user-ID is validated against the file USERID.SYS, an ASCII text file containing entries of the form:

userid, [password], userno[P], [drive]

where "userid" and "password" are up to 8 characters in length, "userno" is a user number 0...30, and "drive" is a drive letter A...P. The password and drive fields are optional. If the given user-ID has an associated password specified in USERID.SYS, then LOGON prompts for a password to be entered and validates it. The log-on succeeds only if both user-ID and password are valid, in which case the console is logged onto the specified user number, and the specified drive is selected as the default disk. The user number suffix "P" in a USERID.SYS entry causes the log-on to be "privileged", enabling the user to access various protected facilities of TurboDOS.

The LOGOFF command sets the user number to a reserved value (normally 31) and selects the system drive as the default disk. The library for user 31 on the system drive normally contains only LOGON.COM and USERID.SYS files. Consequently, no further activity can be performed until a successful LOGON has been accomplished.

If the user 31 library also contains a file named SYSLOG.SYS, then the LOGON and LOGOFF commands will automatically record in that file a chronological log of all log-on and log-off activity.

If the optional module SGLLOG is included, TurboDOS will not permit more than one non-privileged log-on to a particular user number at a time.

Automatic Program Loading

TurboDOS provides a flexible and user-defined facility for loading any program (e.g., a function menu, the LOGON command) automatically at initial system cold-start and/or at each system warm-start. Automatic program loading at cold-start and warm-start are controlled by files named COLDSTRT.AUT and WARMSTRT.AUT respectively. Automatic program loading takes place only if the corresponding .AUT file is present on the logged-on disk.

The AUTOLOAD command is used to create these .AUT files. Alternatively, a .COM file may be automatically loaded simply by renaming it as COLDSTRT.AUT or WARMSTRT.AUT.

The autoload processor is an optional module of TurboDOS, and may be omitted. If the autoload module is omitted, then the command line interpreter module must be included.

File Management

TurboDOS file management is both program and media compatible with CP/M, but it relaxes many of the restrictions of CP/M. TurboDOS supports large hard disks in excess of 1,000 megabytes (65,536 blocks of 16K bytes each) without partitioning. File sizes to 134 megabytes (8,192 extents of 16K bytes each) are allowed for both sequential and random access. Random record numbers are supported to 20-bits (0...1,048,575).

TurboDOS also supports two special pseudo-files. "$.DIR" refers to the directory area of a disk, while "$.DSK" refers to the entire contents of a disk volume (up to a maximum of 134 megabytes). These pseudo-files may be dumped, patched, or accessed like any ordinary file. However, access is restricted to privileged log-ons only.

A third pseudo-file "$.LOK" may be used by any user in connection with shared-file record locks (see below).

File Attributes

Six of the possible eleven file attribute bits (sign bits of the file name/type) in the directory are defined in TurboDOS as follows:

f1 = Exclusive attribute
f2 = Shared attribute
f3 = FIFO attribute
f4-f8 = (unassigned)

t1 = Read-only attribute
t2 = Global attribute ("system file" in CP/M)
t3 = Archived attribute

These attributes may be set by means of (1) a call to TurboDOS function 30 (set file attributes), (2) the SET command, or (3) the normal file renaming facility of many high-level language processors. File attributes may be displayed with the SHOW command.

If the exclusive attribute is set, a file may be opened by only one process at a time; requests by other processes to open the same file are denied (as if the file could not be found). If the read-only attribute is set, a file may not be written, deleted or renamed by any process. If the shared attribute is set, a file may be opened, read, and written by any number of concurrent processes; shared files are also eligible for record locks (described below). If none of these attributes (exclusive, read-only, shared) is set, then a file may be opened and read by multiple processes concurrently, but may be written only by one process.

If the global attribute is set for a file under user number zero, then that file may be accessed from any user number. A global file may be written unless the read-only attribute is also set. The global attribute has no effect for files under non-zero user numbers.

The FIFO attribute indicates that a file is to be accessed using a special first-in-first-out access method (described below).

The archived attribute is set automatically whenever a file is archived by means of the COPY command with the ";A" option (see COPY command), and is automatically reset whenever a file is written.

Setting File Attributes via Rename

To permit file attributes to be set by programs written in high-level languages, TurboDOS includes an extension to its file renaming facility (function 23). If a file is renamed, and if the new name consists solely of the characters "+", "-", "#" and space, then the operation is treated by TurboDOS in a special way. The character "+" causes the corresponding file attribute to be set, while the character "-" causes the attribute to be cleared. Characters "#" and space leave corresponding attributes unchanged. For example, the Microsoft BASIC-80 statement:

```
200   NAME  "DATABASE.DAT"  AS  "#+######.#+#"
```

would have the effect of setting the Shared (f2) and Global (t2) attributes on the file DATABASE.DAT.

File Sharing and Interlocks

TurboDOS provides a user number prefix to file names which permits a disk directory to contain up to 32 logical sub-directories. Most file operations (creating, renaming, deleting, searching, etc.) are restricted to the sub-directory corresponding to the current user number. However, files created under user number zero and given the "Global" attribute may be opened under any other user number. This permits commands, programs, and other common files to be shared by all users.

Multi-user configurations of TurboDOS provide the file and record interlocks necessary to permit simultaneous multi-user access to common data bases. Implicit file-level interlocks are completely automatic, require no user-program participation, and support multiple inquiry tasks concurrent with one update task. Explicit record-level interlocks support concurrent interleaved update by multiple tasks.

The implicit interlocks supported by TurboDOS permit any number of users to read a shared file simultaneously. If any user writes to the file, then that user gains an exclusive write-lock on the file. Attempts by any other user to write to the same file returns an error code. The writing user may extend the file by adding new records at the end, and these records become immediately accessible to all sharing users. The exclusive write-lock is released when the file is closed or the locking program terminates.

Record-Level Interlocks for Shared Files

TurboDOS supports automatic record-level interlocks for files which have the shared attribute set. TurboDOS permits full use of its record-level interlocks from existing high-level language compilers and interpreters (unlike CP/M-3 and MP/M-2). A set of sample programs written in Microsoft BASIC-80 is attached to illustrate the manner in which record-level interlocks may be used in concurrent update applications.

Record locks are controlled by means of a pseudo-file named "$.LOK", which is not a real disk-resident file but rather a means for a program to announce its interlocking intentions to TurboDOS in a CP/M-compatible fashion. Operations on $.LOK (open, create, close, delete, read, write, etc.) involve no disk work and are essentially instantaneous.

A process announces its intention to use record-level interlocks either by opening (function 15) or creating (function 22) the pseudo-file $.LOK. Subsequently, if the process reads a record from a file which has the shared attribute, then the process automatically acquires a lock on that record. Any number of records in any number of shared files may be locked concurrently. Similarly, if the process writes a record to a shared file, then the process automatically releases any lock that it may have on that record. Whenever the process opens, creates, or closes the pseudo-file $.LOK, it relinquishes any outstanding record locks that it may have.

If a process attempts to read or write a record which is locked by someone else, the result depends upon whether the process opened or created $.LOK. If the process opened $.LOK, then any attempts to read or write a locked record cause the process to be suspended until the locked record becomes available. If the process created $.LOK, then any attempts to read or write a locked record are immediately denied, and an error code is returned (A-reg = 8).

TurboDOS allows a shared file to be extended in a concurrent update environment (unlike CP/M-3 and MP/M-2). The extending program should first acquire an interlock by attempting to read past end-of-file (this will return an EOF status, and will acquire an interlock on the record at EOF+1). Then, the program may safely write a new record to EOF+1. If sparse files are used, this same procedure can be used for adding any non-existent record to a random-access file.

FIFOs

To facilitate inter-process and inter-user communications, TurboDOS supports a special kind of file called a "FIFO" (first-in first-out) which is similar to a "pipe" under UNIX. FIFOs are opened, closed, read and written exactly like ordinary sequential files. However, a record written to a FIFO is always appended to the end, and a record read from a FIFO is always taken from the beginning and removed from the FIFO.

A FIFO is differentiated from other files by the presence of the FIFO attribute in the directory. Record zero of a FIFO is used by TurboDOS as a special header record to keep track of the FIFO. The header record specifies whether the body of the FIFO is to be disk-resident or RAM-resident, and the maximum number of records the FIFO may contain. RAM-resident FIFOs provide high speed but limited capacity (up to 127 records, usually much less), while disk-resident FIFOs provide large capacity (up to 65,535 records) but slower speed. The FIFO command may be used to create a FIFO and initialize its header.

Normally, reading from an empty FIFO returns an end-of-file condition (A = 1), and writing to a full FIFO returns a disk-full condition (A = 2). However, reading from an empty FIFO or writing to a full FIFO which has the Shared attribute set causes the process to be suspended until the FIFO becomes non-empty or non-full.

The header or disk-resident body of a FIFO may be accessed directly by using read-random (function 33) and write-random (function 34) operations, thereby bypassing the normal first-in first-out protocol. An attempt to create (function 22) an existing FIFO is treated as an open (function 15), while an attempt to delete (function 19) a FIFO is ignored. The only way to get rid of a FIFO is to first reset the FIFO attribute, then delete it.

Buffer Management

The TurboDOS buffer manager module performs multi-level buffering of disk I/O, using least-recently-used (LRU) buffer assignment and other sophisticated optimizations. Buffering provides a manyfold reduction in the number of physical disk accesses in both sequential and random operations.

The BUFFERS command allows the number and/or size of disk buffers to be changed at any time. The number of buffers must be at least two, and the buffer size must not be smaller than the physical sector size of the disks being used. For optimum performance, the number of buffers should as large as possible consistent with the memory requirements of the programs to be run.

The buffer manager maintains its buffers on two threaded lists: the "in-use" list and the "free" list. Whenever the file manager requests a disk access, the buffer manager first checks the in-use list to see if the requested disk record is already in a buffer. Most of the time it is, and no physical disk access is required. If not, it attempts to acquire a new buffer from the free list. If the free list is empty, the least recently used buffer (at the end of the in-use list) is written out to disk if necessary, and then reused.

Before a removable disk volume is changed, it is crucial that any buffers relating to that disk be written out if necessary, and returned to the free list. In single-user configurations of TurboDOS, this is accomplished automatically whenever the system pauses for console input. In multi-user configurations, buffers are flushed and freed by the CHANGE command (which should be executed before any disk change). Buffers are also flushed automatically during any lull in system activity.

Program Load Optimization

The program load optimizer is an optional TurboDOS module which greatly improves the speed of program loading and overlay fetching. This module scans the allocation map of program files which are to be loaded into memory, determines the sequentially allocated segments of the file (often 16K or more in length), and loads these segments at the maximum transfer rate of the disk controller. This provides a manyfold increase over normal sequential file access performed one record (128 bytes) at a time. The program load optimizer is utilized automatically by the command line interpreter and the autoload processor, and is also accessible to user programs by means of an operating system function call. In networking configurations of TurboDOS, program load optimization is not applicable to programs loaded over the network.

Memory Management

The resident portion of TurboDOS resides in the topmost portion of system memory. TurboDOS uses a common memory management module to provide dynamic allocation and deallocation of memory space required for disk buffers, de-spool requests, file interlocks, DO-file nesting, etc. Dynamic memory segments are allocated downward from the base of the TurboDOS resident area, thereby reducing the space available for the TPA (transient program area). Thus, the size of the TPA can vary slightly from time to time during normal system operation. Deallocated segments are concatenated with any neighbors and threaded on a free list. A best-fit algorithm is used to reduce memory fragmentation.

Networking

TurboDOS supports a multi-user network of interconnected microcomputers which can share a common pool of mass storage, printers and other peripherals. Both master-to-master and master-to-slave interconnections are supported. Slave processors may have their own local disk storage, or may be entirely down-line loaded from a master processor. The network protocol is a simple one, adaptable to both point-to-point and multi-drop links, parallel or serial.

Networking TurboDOS incorporates an advanced failure detection and recovery facility that makes a properly configured master-slave network virtually crashproof. The master processor polls all slave processors periodically. Repeated failure of a slave to respond to a poll is an indication that the slave has crashed. In this case, the master automatically resets and reloads the slave. Furthermore, any open files are automatically closed and any file interlocks are automatically released. No user action is required to effect this recovery. Even a malicious user running in a slave processor cannot compromise the processing of another slave or the files belonging to another user.

A user may intentionally cause a reset and reload of his slave processor by typing two consecutive ASCII unit-separator characters (CTRL-_). A privileged user may reset and reload any slave processor via the RESET command.

Note that this automatic recovery requires a hardware network interface which permits the master processor to interrupt, reset, and down-load a slave processor. Consult Software 2000 Inc. for hardware recommendations.

System Start-Up

TurboDOS uses a start-up technique which does not depend on reserved "system tracks" on each disk. The TurboDOS start-up PROM scans all disk drives, and searches the directories of any ready drives for the loader program "OSLOAD.COM". When this file is found, the start-up PROM loads it into the TPA and executes it.

The loader program scans all disk drives for an executaable version of the operating system "OSMASTER.SYS". When this file is found, the loader program proceeds to load the operating system into the topmost portion of memory, using the load address and load length specified by the first four bytes of the .SYS file.

System Generation

TurboDOS includes an easy-to-use system generator (the GEN command) for creating loaders, executable operating systems, and bootstrap ROMs. The GEN command is a specialized linkage editor which links together the desired combination of functional and hardware-dependent modules to create a version of TurboDOS configured with the desired features and for the desired hardware. The GEN command also provides a powerful symbolic patch facility which may be used to establish virtually any operating system parameter. Consult the Configuration Guide for details.

Example of the Use of TurboDOS Record Locks

The following pages contain listings of three short programs written in Microsoft BASIC-80 which illustrate the manner in which record-level interlocks may be used in concurrent update applications.

The first program, INITIAL.BAS, creates a random data file DATABASE.DAT and initializes all of its data records to the value zero. The first record in the file is a header record, and contains a value which gives the number of data records in the file.

The second program, UPDATE.BAS, performs repetitive random-access updates to the file. Each update increments the value contained in some record by one. Because this program uses the record lock facilities of TurboDOS, it may be executed simultaneously from several terminals at once.

The third program, CHECKSUM.BAS, processes the file sequentially and displays the sum of the values contained in all data records. After running UPDATE.BAS simultaneously from several terminals, CHECKSUM.BAS may be used to verify that all updates were processes correctly.

```
100 ' "INITIAL.BAS" initializes a random database.
110 '
120 INPUT "NUMBER OF RECORDS";N9
130 '
140 ' Create random file
150 OPEN "R",#1,"DATABASE.DAT"
160 FIELD #1,128 AS R$
170 '
180 ' Write header record containing number of data records (N9)
190 LSET R$=STR$(N9)
200 PUT #1,1
210 '
220 ' Write N9 data records initialized to zero
230 FOR N=2 TO N9+1
240 LSET R$=STR$(0)
250 PUT #1,N
260 PRINT N;
270 NEXT N
280 '
290 ' Close the file, and set Shared and Global attributes
300 CLOSE #1
310 NAME "DATABASE.DAT" AS "#+######.#+#"
320 END
```

```
100 ' "UPDATE.BAS" illustrates random-access updating
110 ' using the record lock facilities of TurboDOS.
120 '
130 INPUT "ENTER RANDOM NUMBER";S
140 INPUT "ENTER NUMBER OF ITERATIONS";N9
150 '
160 ' Discard the first S random numbers
170 FOR N=1 TO S:X=RND(1):NEXT N
180 '
190 ' Open random file and read its header record
200 ' to determine the number of data records (R9)
210 OPEN "R",#1,"DATABASE.DAT"
220 FIELD #1,128 AS R$
230 GET #1,1
240 R9=VAL(R$)
250 '
260 ' Generate a random record number between 2 and R9+1.
270 ' Read-increment-write that record, using record locks.
280 ' Repeat this process for N9 iterations.
290 OPEN "I",#2,"$.LOK"   ' enable interlocks (suspend on conflict)
300 FOR N=1 TO N9
310 R=INT(RND(1)*R9)+2
320 GET #1,R
330 X=VAL(R$)
340 X=X+1
350 LSET R$=STR$(X)
360 PUT #1,R
370 PRINT N,R,X
380 NEXT N
390 '
400 ' Finally, close the file
410 CLOSE #1
420 CLOSE #2   ' release interlocks
430 END
```

```
100 ' "CHECKSUM.BAS" simply reads each record
110 ' in the file, and computes the sum of them.
120 ' This program can be used to check for the
130 ' proper operation of "UPDATE.BAS" when used
140 ' concurrently from several terminals.
150 '
160 OPEN "R",#1,"DATABASE.DAT"
170 FIELD #1,128 AS R$
180 GET #1,1
190 N9=VAL(R$)
200 FOR N=2 TO N9+1
210 GET #1,N
220 S=S+VAL(R$)
230 PRINT N;VAL(R$);S
240 NEXT N
250 CLOSE #1
260 PRINT
270 PRINT "SUM =";S
280 END
```

COMMANDS

AUTOLOAD Command

TurboDOS provides a flexible and user-defined facility for loading any program (e.g., a function menu) automatically at initial system cold-start and/or at each system warm-start. Automatic program loading at cold-start and warm-start are controlled by files named COLDSTRT.AUT and WARMSTRT.AUT respectively. Automatic program loading takes place only if the corresponding .AUT file is present on the logged-on disk.

The AUTOLOAD command enables you to create these .AUT files. The command format is:

```
        AUTOLOAD command
```

where "command" is any valid TurboDOS command that you wish to be executed automatically at cold-start or warm-start. The "command" may not be a multi-command string, but it may be a DO-command.

The AUTOLOAD command creates a file named AUTOLOAD.AUT on the logged-on disk. By renaming this file as COLDSTRT.AUT or WARMSTRT.AUT, you will cause the specified "command" to be executed automatically at system cold-start or at each warm-start. Note that a newly-created .AUT file does not affect system operation until the next system cold-start.

Example:

```
        0A}AUTOLOAD MBASIC MENUPROG
        Auto load file created.
        0A}RENAME AUTOLOAD.AUT COLDSTRT.AUT
        A:AUTOLOAD.AUT renamed to A:COLDSTRT.AUT
        0A}
```

BACKUP Command

The BACKUP command enables you to perform a fast disk-to-disk copy for backup purposes. The command format is:

```
BACKUP srcdrive destdrive ;R
```

where "srcdrive" and "destdrive" specify the source and destination drives. Source and destination disks must be of exactly the same type and format. If the ";R" suffix is present, then BACKUP repeats (allowing multiple removable disks to be backed up) until terminated by typing CTRL-C.

In a network configuration, BACKUP may not be used in a slave processor to access source or destination drives owned by the master processor. However, a slave console may be attached to the master processor (see MASTER command) for the purpose of running BACKUP.

Examples:

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

BUFFERS Command

The BUFFERS command enables you to change the number and/or size of disk buffers maintained by TurboDOS to enhance file processing performance. The command format is:

```
BUFFERS Nn Ss
```

where "n" is the desired number of buffers (at least 2), and "s" is the desired buffer size in bytes (128, 256, 512, 1024,..., 16384). The buffer size must not be smaller than the physical sector size of the disks being used. For optimum performance, the number of buffers should as large as possible consistent with the memory requirements of the programs to be run.

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" arguments are omitted, the corresponding parameter remains unchanged. If both are omitted, the command simply displays the current parameters.

Examples:

```
0A}BUFFERS N6
Number of Buffers: 6
Length of Buffers: 1024
Current System Size: 64K
Memory Available: 46596
0A}
```

CHANGE Command

The CHANGE command must be used prior to changing removable disk volumes in multi-user configurations of TurboDOS. The command format is:

```
CHANGE drives
```

where "drives" is a string of one or more drive letters A...P, or "*" if you want to change all disks.

If any of the requested drives 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 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.

Examples:

```
0A}CHANGE BCD
Change disk(s) BCD, then enter carriage-return: <cr>
0A}

0A}CHANGE *
Disk(s) CD in use
0A}
```

COPY Command

The COPY command enables you to copy individual disk files or groups of files. The command format is:

```
COPY srcfile destfile ;options
```

where "srcfile" and "destfile" specify the source and destination drives and files. Wild-card characters ("?" and "*") may be used in the "srcfile" 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 file specification are to be used in specifying each destination file. An "srcfile" or "destfile" argument consisting only of a drive letter (e.g., "C:") implies a file specification of all wild cards (i.e., "C:*.*").

If both "srcfile" and "destfile" are omitted from the command line, then the COPY command operates in an interactive mode. It reads successive directives from the console (prompted by an asterisk "*"). A null directive terminates the command. The format of each interactive directive is:

```
srcfile destfile ;options
```

The "options" argument may contain either "Y" or "N" (preceded by a semicolon), to specify whether or not you want to confirm each individual file before it is copied. If neither "Y" or "N" is specified and "srcfile" contains wild cards, then the COPY command prompts you to specify whether or not confirmation is desired.

The "options" argument may also contain "S" and/or "D" followed by a number from 0 to 31. If present, these options specify the user number associated with the source and/or destination files, respectively. In the absence of these options, the source and destination files are associated with the current user number. The "S" and "D" options are honored for privileged log-ons only.

The "E" option causes each source file to be deleted after it has been successfully copied to the destination.

Two options facilitate archiving of hard disks. The "A" option causes COPY to bypass files which have the Archived attribute set, and to set the Archived attribute on files that are copied. The "C" option gives you the opportunity to change the disk in the destination drive if the first disk becomes full.

For example, the command:

```
0A}COPY E: A: ;ACN
```

could be used to provide incremental archiving of a Winchester hard disk (E:) to several mini-floppies (A:).

Examples:

```
0A}COPY *.BAS B: ;N
A:AMORTIZE.BAS copied to B:AMORTIZE.BAS
A:PRIMES  .BAS copied to B:PRIMES  .BAS
A:STARTREK.BAS copied to B:STARTREK.BAS
0A}

0A}COPY B:MAXI*.* C:OPT*.* ;Y
OK to copy B:MAXICOMP.TXT to C:OPTICOMP.TXT (y/n)? N
OK to copy B:MAXIMUMS.COM to C:OPTIMUMS.COM (y/n)? Y
B:MAXIMUMS.COM copied to C:OPTIMUMS.COM
OK to copy B:MAXIMUMS.FOR to C:OPTIMUMS.FOR (y/n)? Y
B:MAXIMUMS.FOR copied to C:OPTIMUMS.FOR
OK to copy B:MAXIMUMS.REL to C:OPTIMUMS.REL (y/n)? N
0A}

0A}COPY
* AUTOLOAD.COM B:
A:AUTOLOAD.COM copied to B:AUTOLOAD.COM
* *.TXT B: ;NE
A:REFERENC.TXT copied to B:REFERENC.TXT
A:REFERENC.TXT deleted
A:USERGUID.TXT copied to B:USERGUID.TXT
A:USERGUID.TXT deleted
*
0A}
```

DATE Command

The DATE command enables you to set or display the system date and time. The command format is:

```
        DATE SET
```

in response to which you are prompted interactively to enter the new system date (in format "dd mmm yy") and time (in format "hh:mm:ss"). You may leave the date and/or time unchanged simply by typing carriage-return in response to the corresponding prompt. The command:

```
        DATE
```

simply displays the current system date and time.

Examples:

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

DELETE Command

The DELETE command enables you to delete individual disk files or groups of files. The command format is:

```
DELETE file ;option
```

where "file" specifies the file to be deleted. Wild-card characters ("?" and "*") may be used in the "file" argument to indicate that multiple files are to be deleted.

If "file" is omitted from the command line, then the DELETE command operates in an interactive mode. It reads successive directives from the console (prompted by an asterisk "*"). A null directive terminates the command. The format of each interactive directive is:

```
file ;option
```

The "option" argument may be either "Y" or "N" (preceded by a semicolon), and specifies whether or not you want to confirm each individual file before it is deleted. If "option" is omitted and "file" contains wild cards, then the DELETE command prompts you to specify whether or not confirmation is desired.

Examples:

```
0A}DELETE *.BAS ;N
A:AMORTIZE.BAS deleted
A:PRIMES .BAS deleted
A:STARTREK.BAS deleted
0A}

0A}DELETE B:MAXI*.* ;Y
OK to delete B:MAXICOMP.TXT (y/n)? N
OK to delete B:MAXIMUMS.COM (y/n)? Y
B:MAXIMUMS.COM deleted
OK to delete B:MAXIMUMS.FOR (y/n)? Y
B:MAXIMUMS.FOR deleted
OK to delete B:MAXIMUMS.REL (y/n)? N
0A}

0A}DELETE
* AUTOLOAD.COM
A:AUTOLOAD.COM deleted
* *.TXT
Ambiguous filename: confirm individual files (y/n)? N
A:REFERENC.TXT deleted
A:USERGUID.TXT deleted
*
0A}
```

DIR Command

The DIR command enables you to display an alphabetized disk directory on the console or printer. The command format is:

```
DIR file ;L
```

If the "file" argument is present, it specifies the desired disk drive. Otherwise, the logged-on drive is referenced. The "file" argument may also include an ambiguous file specification to indicate that only a specified group of files is to be displayed.

If the ";L" suffix is present, then the directory is printed. Otherwise, it is displayed on the console.

The DIR command displays only files created under the current user number. (Also see the USER command.)

Examples:

```
0A}DIR ;L
...directory of all A-drive files on printer...
0A}

0A}DIR B:
...directory of all B-drive files on console...
0A}

0A}DIR *.BAS
...directory of A-drive .BAS files on console...
0A}
```

DO Command

The DO command enables you to execute a pre-defined sequence of TurboDOS commands which you have previously placed in a disk file. The command format is:

```
        DO dofile arg1 arg2 ... argN
```

where "dofile" specifies a file of commands to be executed (called the "DO-file"). If "dofile" does not specify an explicit file type, then the type ".DO" is assumed. If the DO command has no additional arguments following "dofile", then the commands in the specified DO-file are simply executed by TurboDOS in sequence.

The optional arguments "arg1" through "argN" are arguments to be substituted into marked locations in the DO-file. Any number of arguments is permitted. Arguments containing embedded spaces must be enclosed in single or double quotes. If one or more of these arguments are present, then the DO-command makes a temporary copy of the DO-file in which the arguments are substituted as required. The commands in the temporary file are then executed by TurboDOS in sequence. The temporary file is given the same name as the original DO-file except that the last character of the file type is changed to a dollar sign (e.g., "GENERATE.DO" is copied to "GENERATE.DO$"). The last line of the temporary file consists of a DELETE command which causes the temporary file to be deleted at the end of execution.

The DO-file is simply an ASCII file of any desired length, each line of which contains a valid TurboDOS command or multi-command string. Thus, you can create DO-files with any conventional text editor. 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}" will be replaced by the value of the argument "arg3". A default value may follow the argument number, separated by a comma (e.g., "{12,DEFAULT}"), and will be used if the corresponding argument of the DO command is missing or null.

A DO-file may contain any number of embedded DO commands, even in multi-command strings. TurboDOS supports full nesting of DO-files to any reasonable depth.

Certain commands (such as COPY, RENAME and DELETE) and other programs expect interactive input from the console keyboard. If such a command or program is executed within a DO-file, then its console input comes from the DO-file rather than the keyboard. An exception is console input solicited by means of direct console I/O (function 6) or direct calls on the BIOS vector.

Examples:

Suppose you have created a file named RUNBASIC.DO containing the following lines of ASCII text:

```
BASCOM {1}.REL,{1}.PRN={1}.BAS{3,/Z/S/C}
TYPE {1}.PRN ;L
L80 {1}.REL{2}/E,{1}.COM/N
DELETE
{1}.REL
{1}.PRN
<null line>
{1}
```

Then you could execute this DO-file as follows:

```
0A}DO RUNBASIC PRIMES /M
0A}BASCOM PRIMES.REL,PRIMES.PRN=PRIMES.BAS/Z/S/C
...compilation...
0A}TYPE PRIMES.PRN ;L
...listing on printer...
0A}L80 PRIMES.REL/M/E,PRIMES.COM/N
...link map...
0A}DELETE
* PRIMES.REL
A:PRIMES .REL deleted
* PRIMES.PRN
A:PRIMES .PRN deleted
*
0A}PRIMES
...execution of primes program...
0A}DELETE RUNBASIC.DO$
A:RUNBASIC.DO$ deleted
0A}
```

DRIVE Command

The DRIVE command enables you to display physical disk characteristics on the console or printer. The command format is:

```
DRIVE file ;L
```

If the "file" argument is present, it specifies the desired disk drive. Otherwise, the logged-on drive is referenced. If the ";L" suffix is present, then the drive information is printed. Otherwise, it is displayed on the console.

Examples:

```
0A}DRIVE B:
Disk characteristics, drive B:DOCUMENT.TXT
Maximum data capacity        : 1224K
Allocation block size        : 2K
Number of directory entries  : 256
Physical sector size         : 1024
Physical sectors per track   : 16
Physical tracks per disk     : 77
Number of reserved tracks    : 0
0A}
```

DUMP Command

The DUMP command enables you to display a combined hex/ASCII file dump on the console or printer. The command format is:

```
DUMP file ;L
```

where "file" specifies the disk file to be dumped. If the ";L" suffix is present, then the hex/ASCII dump is printed. Otherwise, it is displayed on the console.

Examples:

```
0A}DUMP B:DUMP.COM
...combined hex/ASCII dump...
0A}
```

ERASEDIR Command

The ERASEDIR command enables you to erase the entire directory on a specified disk. The command format is:

```
ERASEDIR drive
```

where "drive" specifies the disk to be erased.

CAUTION: This command erases all information on the specified disk, regardless of user number or read-only attributes. Its use is restricted to privileged log-ons only.

Examples:

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

FIFO Command

The FIFO command enables you to create a disk-resident or RAM-resident FIFO. The command format is:

```
FIFO file
```

where "file" specifies the desired drive and name. If the specified FIFO already exists, its characteristics are displayed. Otherwise, you are prompted 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
Enter maximum number of records (1-65535): 1000
FIFO file created
0A}

0A}FIFO B:BATCH.DO
FIFO is Disk resident
Maximum number of records: 1000
Current number of records: 0
0A}
```