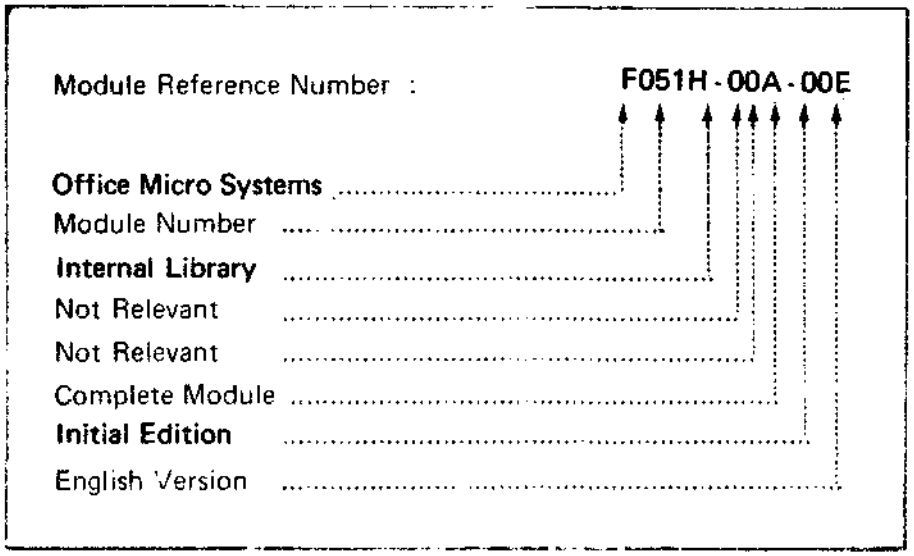
P3500

TURBODOS user's guide for programmers

F51H internal library

Data Systems

PHILIPS



A publication of

Philips Data Systems

SSS - Training & Documentation

Apeldoorn, The Netherlands

Order Number 5122 993 62831

User's Guide to TurboDOS 1.2

May, 1982

TABLE OF CONTENTS

SECTION 3 -- COMMANDS

SECTION 4 — CP/M-COMPATIBLE FUNCTIONS

SECTION 5 -- ADDITIONAL TURBODOS FUNCTIONS

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.

Available Configurations

Networking TurboDOS supports a multi-user network of interconnected Z80 microcomputers which can share a common pool of mass storage, printers and other pheripherals. Since this distributed processing approach provides a microcomputer dedicated to each user, Networking TurboDOS is able to support a large number of simultaneous users with excellent performance and minimal interaction. Presently, there are a large number of multi-processor TurboDOS installations in the 4- to 8-user range (and some installations up to 16 users), all experiencing excellent response and throughput. Comparative benchmarks have shown that a multi-processor TurboDOS system can outperform a time-shared minicomputer in the PDP-11/34 class at less than half the hardware cost.

Single-user configurations of TurboDOS are also available, both with and without concurrent print spooling capability. Single-user TurboDOS is a superior Z80 replacement for Digital Research CP/M, providing much better performance, increased diskette capacity, enhanced reliability, and numerous functional features not available in CP/M.

TurboDOS is not available in a "time-sharing" configuration (multiple users sharing a single processor). Although TurboDOS includes full multi-tasking facilities, Software 2000 has elected not to release a time-sharing version because of the severe performance and reliability problems associated with time-sharing of an 8-bit microprocessor (well known to users of MP/M).

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 compatible with CP/M 3 and MP/M II in the areas of file and record locking. 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. TurboDOS incorporates a system generation facility which links and relocates the appropriate operating system modules. The system generation facility also incorporates a symbolic patch facility which makes it easy to alter dozens of operating system parameters.

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) ignore 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. The spooler handles the print stream in a transparent fashion, permitting complex print operations such as proportional spacing and micro-space justification to be fully supported. Up to 16 concurrent printers and up to 16 separate print queues are supported. Even in a system with only one printer, the multi-queue capability of the TurboDOS spooler allows documents with different forms requirements to be segregated conveniently.

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. File and record interlocks in TurboDOS provide compatibility with MP/M II, but incorporate numerous enhancements and extensions which alleviate the inadequacies of the MP/M interlock scheme. 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, Networking TurboDOS is able to support a large number of simultaneous users with excellent performance. Networking TurboDOS is especially well-suited to the new generation of multi-processor microcomputer hardware which is now available from many manufacturers. 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. Presently, there are a large number of multi-processor TurboDOS installations in the 4- to 8-user range (and some installations up to 16 users), all experiencing excellent response and throughput. Comparative benchmarks have shown that a multi-processor TurboDOS system can outperform a time-shared minicomputer in the PDP-11/34 class at less than half the hardware cost.

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, or has a TurboDOS bootstrap recorded on reserved disk tracks, 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 no provision for automatically loading TurboDOS, 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
- Network manager
- Multi-task dispatcher
- 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
- Print spooler
- Print 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.

The Configuration Guide describes in detail the various operating system modules and how they may be combined to produce various TurboDOS configurations. The same document describes how to implement TurboDOS device drivers for specific hardware environments.

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 all BDOS functions of CP/M Version 2.2:

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*	Raw Console Input	23	Rename File
4*	Raw Console Output	24	Return Login Vector
5	Output Character to Printer	25	Return Default Disk
6	Direct Console I/O	26	Set Record Buffer Address
7	Return I/O Byte	27*	(Return Disk ALV Address)
8	Set I/O Byte	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 DPB Address
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 compatible with the corresponding functions in CP/M Version 2.2, except for the four functions marked with an asterisk above. Function 3 (Raw Console Input) and function 4 (Raw Console Output) are compatible with MP/M-II, and replace the Reader and Punch functions of CP/M. Function 27 (Return Disk ALV Address) performs no operation in TurboDOS, but this function affects only the CP/M "STAT" utility which is not normally used with TurboDOS. Function 40 (Write Random with Zero Fill) is treated by TurboDOS as a synonym for function 34 (Write Random). Refer to Section 4 for a detailed description of each function.

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).

MP/M-Compatible Functions

In addition to BDOS functions 0-40 supported by CP/M Version 2.2, MP/M-II and CP/M-3 have implemented additional functions 41-48 and 100-107. TurboDOS provides compatible support for certain of these functions, but not for others:

41*	(Test & Write Random)	100*	(Set Directory Label)
42	Lock Record	101*	(Return Dir. Label Byte)
43	Unlock Record	102*	(Read XFCB)
44*	(Set Multi-Sector Count)	103*	(Write XFCB)
45*	(Set BDOS Error Mode)	104	Set Date and Time
46	Get Disk Free Space	105	Return Date and Time
47	Chain to Program	106*	(Set Default Password)
48*	(Flush Buffers)	107	Return BDOS Serial Number

The functions marked with an asterisk above perform no operation under TurboDOS, while the remaining functions are supported in fashion compatible with MP/M-II and CP/M-3. In general, TurboDOS does honor MP/M-II functions concerned with record locking and system date/time, but does not support file passwords, multi-sector count, test-and-write, or extended error mode. Refer to Section 4 for a detailed description of each function.

Several CP/M- and MP/M-compatible functions which are not often used (functions 7, 8, 24, 28, 29, 31, 37, and 107) are implemented as a separate module (CPMSUP) which may be omitted from TurboDOS system generation to reduce the size of the operating system.

Additional TurboDOS Functions

In addition to the CP/M- and MP/M-compatible functions listed above, TurboDOS supports more than 50 additional functions. These are numbered in the range 64-99 and 108-127 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, 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 Version 2.2. 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 Version 2.2, 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 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 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:

PRINT DRIVE=d QUEUE=q 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).

PRINT DRIVE=d FILE causes print output to be spooled to print files on the specified drive (d=A...P). The print files are not 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.

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

PRINT CONSOLE causes print output to be routed to the console.

PRINT OFF causes print output to be discarded.

In a multi-user Networking TurboDOS configuration, each user may control his own print routing independently. Various printers, queues and drives may belong to the user's own processor or to any other processor in the network.

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.

In a multi-user Networking TurboDOS configuration, a user may control de-spooled printing on any printer, whether it belongs to the user's own processor or to another processor in the network.

The spooler and de-spooler are optional modules of TurboDOS, and may be omitted in systems where memory space is at a premium. In most multi-user systems, however, print spooling is an operational necessity.

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.

File Attributes

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

f1 = FIFO attribute
f2-f4 = (unassigned, available to user)
f5-f8 = (interface attributes, reserved)

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

These attributes may be set with the TurboDOS SET command, or via a call to TurboDOS function 30 (set file attributes). File attributes may be displayed with the SHOW command.

If the read-only attribute is set, a file may not be written, deleted or renamed by any process.

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

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.

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

The interface attributes are used during file open, create and close functions to determine the open mode of a file (also described below).

File Libraries and User Numbers

On each disk volume, TurboDOS provides 32 separate file libraries corresponding to user numbers 0...31. Generally, user number 0 is reserved as the global system library and user number 31 is reserved for logon security, leaving user numbers 1...30 for general use. The TurboDOS logon procedure establishes the user number for each non-privileged user until the user logs off. Privileged users may change from one user number to another without restriction.

TurboDOS treats the user number as a prefix to file names, thereby allowing 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.

File-Level Interlocks for Shared File Access

Multi-user configurations of TurboDOS provide the file and record interlocks necessary to permit simultaneous multi-user access to common data bases. File-level interlocks are controlled by interface attributes at the time files are opened or created, and support four different modes of file access. Record-level interlocks are controlled by explicit locking and unlocking function calls, and support concurrent interleaved update by multiple tasks. These interlock facilities are compatible with MP/M-II, but provide numerous extensions to alleviate the most serious deficiencies in the MP/M-II file sharing scheme (see discussion of compatibility modes below).

TurboDOS provides four different modes of opening a file—exclusive, shared, read-only, and permissive—defined as follows:

Exclusive: A file opened in exclusive mode is "owned" by the opening process exclusively until it is closed, and may not be opened by any other process. TurboDOS does not permit a file to be opened in exclusive mode if the file is currently opened (in any mode) by another process.

Shared: A file may be opened in shared mode by any number of processes simultaneously, and all processes are allowed to read, write and extend the file freely. If a process extends the file by adding new records at the end, these records become immediately accessible to the other processes that also have the file opened. TurboDOS record lock and unlock functions are honored only for files opened in shared mode.

Read-Only: A file may be opened in read-only mode by any number of processes simultaneously, and all processes are allowed to read the file, but not to write or extend it.

Permissive: A file may be opened in permissive mode by any number of processes simultaneously, and all processes are allowed to read the file. If any process writes or extends the file, then that process gains an exclusive write-lock on the file. Attempts by any other process to write to the file returns an error code. If a process extends the file by adding new records at the end, these records become immediately accessible to the other processes that also have the file opened. The exclusive write-lock is released when the locking program closes the file or terminates.

The open mode of a file is determined by the use of interface attributes f5 and f6 in the file control block at the time the file is opened or created. See Section 4 for details.

Record-Level Interlocks for Concurrent File Update

TurboDOS supports record-level interlocks for files which are opened in the shared mode, allowing simultaneous update by multiple processes. TurboDOS functions 42 (lock record) and 43 (unlock record) provide an explicit means for programs to coordinate their activities in a concurrent update environment. These functions are implemented in a fashion which is compatible with MP/M-II, but with significant extensions.

Record-level interlocks are by no means automatic, and require explicit cooperative participation by all updating programs. A programs must lock a record before reading it, and must unlock the record after updating it. If a program attempts to lock a record that is already locked by another process, the lock record function returns an error return code to the program, and the program must try again until successful. Alternatively, TurboDOS can be instructed to automatically suspend program execution until the lock request can be satisfied.

TurboDOS allows a file opened in the shared mode to be extended in a concurrent update environment. The extending program should first acquire a lock on the record EOF+1 (where EOF is the last record in the file). The program may then safely write a new record EOF+1, and finally unlock EOF+1. If sparse files are used, this same procedure can be used for adding any non-existent record to a random-access file.

File Sharing Compatibility Modes

The file and record interlock facilities of TurboDOS are designed to provide compatibility with MP/M-II, yet at the same time to alleviate the most serious limitations of MP/M file sharing. TurboDOS may be instructed to adhere strictly to MP/M file sharing rules, or alternatively to relax one or more of these rules. To this end, TurboDOS provides a byte of "compatibility flags" with the following bit assignments:

bit 7 = Permissive
bit 6 = Suspend
bit 5 = Global Write
bit 4 = Mixed Mode
bit 3 = Logical
bits 2-0 = (not defined)

For each compatibility flag, a zero-bit denotes strict adherence to an MP/M rule, while a one-bit signifies a relaxation of that rule. The initial setting of the compatibility flags may be established during TurboDOS system generation by assigning the desired value to the symbolic location "COMPAT". A program may modify the compatibility flags via TurboDOS function 120 (set compatibility flags), but the flags automatically revert to their initial setting at each program termination.

If the "permissive" flag (bit 7) is set to one, then the default file open mode is permissive. Otherwise, it is exclusive (as in MP/M). Specifically, the file open mode is determined by the interface attributes f5 and f6 in the FCB of an open or create function as follows:

Permissive Flag	f6	f5	Open Mode
0	0	0	exclusive
0	0	1	shared
0	1	0	read-only
0	1	1	permissive
1	0	0	permissive
1	0	1	shared
1	1	0	read-only
1	1	1	exclusive

If the "suspend" flag (bit 6) is set to one, then an attempt to lock a record that is already locked by someone else causes the process to be suspended (blocked) until the lock request can be satisfied. Otherwise, an attempt to lock or write to a record that is already locked by someone else results in an immediate error status return (as in MP/M).

If the "global write" flag (bit 5) is set to one, then programs running under a non-zero user number may access global files on a read/write basis. Otherwise, access to global files is strictly read-only (as in MP/M).

If the "mixed mode" flag (bit 4) is set to one, then one program may open a file in shared mode while another has it open in read-only mode (or vice-versa). Otherwise, the shared mode and the read-only mode are mutually exclusive (as in MP/M).

If the "logical" flag (bit 3) is set to one, then the FCB random record number field for functions 42 (lock record) and 43 (unlock record) is interpreted as an arbitrary 24-bit logical record number which is not validated and does not cause file positioning. Otherwise, the FCB random record number field for functions 42 and 43 is interpreted as the relative record number of a 128-byte record, and positions the file to that record (as in MP/M).

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, and is maintained in the following format:

```
.BYTE   DISK    ;FIFO type (-1 = disk, 0 = RAM)
.BYTE   SUSP    ;FIFO mode (-1 = suspend, 0 = return error)
.WORD   MAXSIZ  ;FIFO maximum size (records)
.WORD   CURSIZ  ;FIFO current size (records)
.WORD   INPREC  ;FIFO last record number read
.WORD   OUTREC  ;FIFO last record number written
.BYTE   [118]0  ;not used, reserved
```

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, if the "suspend indicator" in the FIFO header is set, then reading from an empty FIFO or writing to a full FIFO 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.

Disk Error Handling

In the event of an unrecoverable disk error detected by a device-dependent disk driver, TurboDOS displays a diagnostic message in one of the following formats:

```
Read Error, Drive A, Track 0, Sector 2 [Retry, Ignore, Abort]
Write Error, Drive B, Track 5, Sector 16 [Retry, Ignore, Abort]
Not Ready Error, Drive C [Retry, Abort]
```

and waits for the operator to chose a desired error recovery option by keying the appropriate letter "R", "I" or "A".

The messages "Read Error" and "Write Error" indicate that a read or write operation could not be successfully completed even after a (driver-defined) number of retries. Because the TurboDOS buffer manager optimizes disk write operations by deferring them as long as possible, disk write errors may be reported later than expected, and possibly even to a different user than expected.

The message "Not Ready Error" means either that there is a not-ready hardware condition on the selected drive (e.g., because no disk is mounted), or that the format of the disk is not recognizable.

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.

To provide for dynamic memory allocation while a program is running in the TPA, a safety margin called "memory reserve" is provided between the lower bound of dynamic space and the upper bound of the TPA. The size of this memory reserve may be defined during system generation.

Networking Capabilities

TurboDOS supports a multi-user network of interconnected microcomputers which can share a common pool of mass storage, printers and other peripherals. A processor on the network may have its own local console, printer, and/or mass storage, or may rely completely on peripherals attached to other processors on the network. Even the TurboDOS operating system may be loaded entirely over the network from another processor. The network protocol is a simple one, adaptable to both point-to-point and multi-drop links, parallel or serial, polled or autonomous. However, a data rate between 100 kilobits and 1 megabit per second is recommended for satisfactory network performance.

TurboDOS supports both unidirectional (master-slave) and bidirectional (master-master) dialogue between processors on the network. For example, each of two processors on the network may simultaneously access disks attached to the other.

Networking TurboDOS incorporates an advanced failure detection and recovery facility that makes a properly configured master-slave network virtually crashproof. Typically, 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.

Note that this automatic recovery requires a hardware network interface which permits the master processor to interrupt, reset, and down-load a slave processor. A considerable variety of off-the-shelf microcomputer hardware presently exists which meets all requirements for an efficient TurboDOS network. Consult your TurboDOS distributor or Software 2000 Inc. for hardware recommendations.

Network Architecture

TurboDOS accomodates a wide variety of network topologies, ranging from the simplest master-slave systems to the most complex star, ring and hierarchical networks. A TurboDOS network is defined to consist of 1-255 "circuits", with 2-255 "nodes" (i.e., processors) on each circuit. Each node has a unique 16-bit network address consisting of an 8-bit circuit number plus an 8-bit node number on that circuit. A processor may be attached to several circuits, if desired. A processor which is attached to multiple circuits has multiple network addresses, one for each circuit. Such a processor even may be set up to perform message forwarding, in order to permit dialogue between nodes on different circuits.

The network topology is defined by tables in each processor which are established during TurboDOS system generation. These tables include:

NMBCKT is a byte value which defines the number of network circuits to which this processor is attached.

CKTAST is the circuit assignment table, which defines the network address by which this processor is known on each circuit, and specifies the network circuit driver module responsible for each circuit number.

DSKAST is the disk assignment table, which specifies for all disk drives (A...P) which ones are local to this processor, and which are remote. This table specifies the network address of the processor which owns each remote drive, and the disk driver module responsible for each local drive.

PTRAST is the printer assignment table, which specifies for all printers (A...P) which ones are local to this processor, and which are remote. This table specifies the network address of the processor which owns each remote printer, and the printer driver module responsible for each local printer.

QUEAST is the queue assignment table, which specifies for all de-spool queues (A...P) which ones are local to this processor, and which are remote. This table specifies the network address of the processor which owns each remote queue.

DEFDID is the default network destination ID, used for routing all network requests which are not related to a disk drive, printer, or de-spool queue.

FWDTBL is the message forwarding table which specifies any additional circuits (not directly connected to this processor) which may be accessed via explicit

message forwarding, and how messages destined for such circuits are to be routed.

These tables are quite easy to set up for simple master-slave networks. For complex multi-circuit networks, the set-up is more complicated (as might be expected). Refer to the Configuration Guide for system generation details.

Network Message Forwarding

TurboDOS networking supports two types of message forwarding: implicit and explicit. To understand how each of these works, consider the simple example of three processors (P1...P3) connected by two circuits (C1 and C2) as follows:

P1 <——C1——> **P2** <——C2——> **P3**

Assume a program running in processor P1 accesses drive "D". If P1's DSKAST specifies that P1's "D" is a remote drive defined as P2's "B", then P1 will make a network request via circuit C1 to processor P2. Now suppose that P2's DSKAST specifies that P2's "B" is a remote drive defined as P3's "A". Then P2 will in turn make a network request via circuit C2 to processor P3. Thus, P1 has implicitly accessed a drive belonging to P3, although P1 was not even aware of the existence of P3. This is called implicit forwarding.

Alternatively, if P1's DSKAST specifies that P1's "D" is a remote drive defined as P3's "A", and if P1's FWDTBL specifies that messages destined for circuit C2 may be sent via circuit C1, then P1 will make a network request via circuit C1 to processor P3. Processor P2 will receive the request, recognize that it should be forwarded, and retransmit the request to processor P3 via circuit C2. Thus, P1 has explicitly accessed a drive belonging to P3 with the help of P2, although P1 was not even aware of the existence of P2. This is called explicit forwarding.

System Start-Up

TurboDOS uses a start-up technique which does not depend on reserved "system tracks" on each disk. The TurboDOS start-up ROM 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 ROM loads it into the TPA and executes it.

In some hardware implementations, it may be desirable not to install a special TurboDOS start-up ROM. In such cases, the TurboDOS start-up code may be recorded on system tracks instead.

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.

The disk drive from which "OSMASTER.SYS" is loaded becomes the "system disk", and is used in a master-slave network for downloading TurboDOS into slave processors. Consequently, it is advisable not to change this system disk during multi-user operation. If the master processor has a Winchester disk or other fixed-media storage, it should be used as the system disk.

System Generation Facility

TurboDOS includes an extremely versatile system generation facility (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 the values of dozens of operating system parameters. Consult the Configuration Guide for details.

Adaptation to Other Languages

TurboDOS has been designed to allow easy adaptation to other languages. All messages in the operating system and command processors have been segregated into separate modules, and source code for these message modules is available to TurboDOS OEMs with language requirements other than English.

(Intentionally left blank)

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.

The use of this command is restricted to privileged log-ons only. In a network configuration, BACKUP may not be executed in one processor to access source or destination drives attached to another processor. However, a slave console may be attached to the master processor (see MASTER command) for the purpose of running BACKUP.

Example:

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

BATCH Command

The BATCH command provides a convenient method of entering command strings into a FIFO for processing by a dedicated batch processor in a networking system. The command format is:

BATCH commandstring

The argument may be any valid TurboDOS command or a string of commands separated by a special delimiter, the vertical bar character "|".

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 zero on drive A, and should have the "suspend indicator" set in the FIFO header (see FIFO command). You must also have a dedicated batch processor which processes this FIFO by means of the AUTOLOADed command "DO BATCH". Thus, any command string which is written to the FIFO will wake up the batch processor and be executed in that processor. Whenever the FIFO runs empty, the batch processor will be suspended awaiting additional work to do.

The BATCH command writes the "commandstring" to BATCH.DO, converting each vertical bar "|" to a backslant "\", and prefixing the command string with commands to log onto the proper disk and select the proper user number.

Example:

```
5C}BATCH PASM DSKFLEX X | RELCVT DSKFLEX
5C}
```

would write the following data to the FIFO "BATCH.DO":

USER 5 \ C: \ PASM DSKFLEX X \ RELCVT DSKFLEX

NOTE: The command processor BATCH.COM has two patchable locations. Location 103H contains the command separator character and defaults to "|". Location 104H contains the drive on which BATCH.DO is located and defaults to 0 (drive A). The MONITOR command may be used to patch these paramaters.

BOOT Command

The BOOT command enables you to access any reserved tracks ("boot tracks") on a disk. The command format is:

BOOT source destination

where "source" and "destination" may be either a file specification or a drive specification. If either "source" or "destination" is a drive specification (i.e., a drive letter A...P followed by a colon ":"), it references the reserved track area on that drive.

Note that the BOOT command reads the entire source file or tracks into the TPA before writing the data out to the destination file or tracks. If the source is too big to fit into the TPA, the BOOT command will so indicate. Also, the BOOT command will diagnose a drive specification if the disk in that drive has no reserved tracks.

Example:

```
0A}BOOT B: BOOTSAVE.SYS
Reading from reserved tracks...
Writing to file...
0A}BOOT BOOTCODE.SYS B:
Reading from file...
Writing to reserved tracks...
0A}BOOT B: A:
Reading from reserved tracks...
Writing to reserved tracks...
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.

The use of the BUFFERS command to change buffer parameters is restricted to privileged log-ons only. In a network configuration, 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
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.

The "X" option causes the copy operation not to be performed if the destination file already exists.

Several 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:).

If the hard disk contains a file so large that it cannot fit on a single destination floppy disk, then the "B" option must be utilized to such a file to be copied in sections to multiple destination disks. If this option is used during backup, then it must also be used to recover the file from the backup disks. Furthermore, the operator must take special care during recovery to mount the disks in exactly the same sequence used during the backup operation. Finally, the "B" option is not allowed if "srcfile" contains wild cards.

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.

Example:

```
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}
```

NOTE: The command processor DIR.COM has two patchable locations. Location 103H contains the left margin used in printing the directory (";L" option) and defaults to 3. Location 104H contains the character used to clear the console screen on multi-page displays and defaults to 0CH (ASCII form-feed). The MONITOR command may be used to patch these paramaters.

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.

Example:

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 drive ;L

If the "drive" 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.

Example:

```
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.

Example:

```
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.

Example:

```
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.DQ
FIFO file not found, creating new file
Enter FIFO type (Ram/Disk): D
Suspend processing on FIFO empty or full (Y/N): Y
Enter maximum number of records (1-65535): 1000
FIFO file created
0A}

0A}FIFO B:BATCH.DQ
FIFO is Disk resident
Processing is suspended on FIFO empty or full
Maximum number of records: 1000
Current number of records: 0
0A}
```

FIXMAP Command

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 (e.g., failure to close a newly-created file) 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 enables you to regenerate the allocation map for a disk, thereby reclaiming any disk blocks that may have become unavailable in this fashion. The command format is:

```
FIXMAP drive
```

If the "drive" argument is present, it specifies the desired disk drive. Otherwise, the logged-on drive is referenced.

In a multi-user system, you cannot use FIXMAP on a drive that is in use by another user.

Example:

```
0A}FIXMAP B:
Drive B disk map re-initialized
0A}FIXMAP C:
Drive C disk map re-initialized, 3 allocation block(s) gained
0A}
```

FORMAT Command

The FORMAT command enables you to initialize or verify a disk. The command format is:

FORMAT drive ;options

where "drive" specifies the disk to be initialized. The command normally initializes and verifies the disk, but the ";V" option may be used to cause verification only. If the ";R" option is present, then FORMAT repeats (allowing multiple removable disks to be initialized or verified) until terminated by typing CTRL-C.

Other options on the FORMAT command are hardware-dependent. For floppy disks, the following options are available:

1 or 2	one- or two-sided diskette
S or D	single- or double-density
T or C	TurboDOS or CP/M-compatible format
3, 4 or 8	For minifloppies: 35-, 40- or 80-track disks

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

Example:

```
0A}FORMAT B: ;2DT
Insert disk to be formatted in drive B
Enter <cr> to begin formatting: <cr>
Starting format pass
. . . . . .
Starting verify pass
. . . . . .
0A}
```

NOTE: On systems with several types of disk storage units, several hardware-dependent versions of the FORMAT command (called FMTxxx) will generally be provided.

LABEL Command

The LABEL command enables you to record a volume label on any disk. The command format is:

LABEL label

The "label" argument has the same format as a file specification. The name and type fields are used as a volume label for the specified drive. If no drive is specified explicitly, then the logged-on drive is labelled. The label recorded on a disk may be displayed using the DIR or DRIVE command.

Example:

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

LOGON Command

The LOGON command provides password-type security in multi-user configurations of TurboDOS. The purpose of this command is to prevent unauthorized access to the system and to protect private file libraries. The command format is:

LOGON

in response to which you are prompted to enter your user-ID from the console keyboard. The user-ID is validated against the file USERID.SYS in the user 31 library. USERID.SYS is 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 your user-ID has an associated password specified in USERID.SYS, then LOGON prompts you to enter a password, and validates it. The log-on succeeds only if you enter both the user-ID and password correctly, in which case your console is logged onto the specified user number, and the specified drive is selected as the default disk. If your entry in USERID.SYS has the user number suffix "P", you are logged-on as a "privileged" user, enabling you to access various protected facilities of TurboDOS.

If the user 31 library also contains a file named SYSLOG.SYS, then the LOGON command will automatically record your log-on in that file.

NOTE: The command processor LOGON.COM has a patchable byte at location 103H which contains the character used to clear the console screen and defaults to 0CH (ASCII form-feed). The MONITOR command may be used to patch this paramater.

LOGOFF Command

The LOGOFF command is used in multi-user configurations of TurboDOS to terminate your session. The command format is:

LOGOFF

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

If the library for user 31 on the system drive also contains a file named "SYSLOG.SYS", then the LOGOFF command will automatically record your log-off in that file.

Example:

```
5C]LOGOFF
31A]
```

MASTER Command

If you are using a slave console in a networking TurboDOS configuration, the MASTER command enables you to attach your console to the master processor. The command format is:

MASTER

which attaches your console to the master processor. To detach from the master processor (and resume normal slave console operation), enter an Attention-Abort sequence (CTRL-S CTRL-C).

While attached to the master, you can make attention requests of the master processor by using CTRL-A (instead of the usual CTRL-S).

NOTE: The MASTER command accesses the master processor as defined by DEFDID in the slave, and requires that the master operating system be generated with a special console driver module (CONREM).

Example:

```
3B}MASTER
Console attached to master processor
Enter User-ID: SYSTEM
Enter Password: SECRET
0A}BACKUP A: B:
.
.
.
0A}LOGOFF
Enter User-ID: <CTRL-S CTRL-C>
Console detached from master processor
3B}
```

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,addr2[,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 if no 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,addr2] 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 descrepancies 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
* E 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}
```

PRINT Command

The PRINT command enables you to control the routing of your print output. The command format is:

PRINT keyword keyword...

where "keyword" is chosen from the following list:

PRINTER=p
QUEUE=q
DRIVE=d
FILE
CONSOLE
OFF

and "p", "q" and "d" are printer, queue and drive letters in the range A...P. Any keyword may be abbreviated to a single letter ("P" for PRINTER, "Q" for QUEUE, etc.).

PRINT DRIVE=d QUEUE=q causes print output to be spooled to print files on the specified drive, then queued automatically on the specified print queue for de-spooled printing.

PRINT DRIVE=d FILE causes print output to be spooled to print files on the specified drive, but not queued automatically for printing.

PRINT PRINTER=p causes print output to be routed directly to the specified printer without intermediate spooling to disk.

PRINT CONSOLE causes print output to be routed to the console.

PRINT OFF causes print output to be discarded.

PRINT without keywords causes the current print routing to be displayed.

Examples:

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

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

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

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

PRINTER Command

The PRINTER command enables you to control de-spooled printing. The command format is: .

PRINTER p keyword keyword...

where "p" is a printer letter in the range A...P which specifies the printer to be controlled, and "keyword"is chosen from the following list:

QUEUE=q
OFFLINE
STOP
GO
BEGIN
TERMINATE

Any keyword may be abbreviated to a single letter ("T" for TERMINATE, "Q" for QUEUE, etc.).

PRINTER p QUEUE=q causes the specified printer to take its print jobs from the specified de-spool queue. 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.

PRINTER p with no keywords displays the current status of the specified printer.

Examples:

```
0A}PRINTER B Q=A
Printer B Assigned to QUEUE A
0A}
```

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

QUEUE Command

The QUEUE command enables you to manually queue print files for de-spooled printing. The command format is:

QUEUE file ;options

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

If "file" is omitted from the command line, then the QUEUE 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 ;options

The options argument may include:

"Y" or "N" to specify whether or not you want to confirm each individual file before it is queued. If neither is specified and "file" contains wild cards, then the QUEUE command prompts you to specify whether or not you want confirmation.

"S" or "D" to specify whether you want the queued files to be saved or deleted after printing. If neither is specified, then the files are saved.

"Q=q" (where is a queue letter in the range A...P) to specify which de-spool queue to use. If not specified, the current queue (as specified in the last PRINT command) is used. If there is no current queue in effect, then the "Q=q" option is required.

Example:

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

RECEIVE Command

The RECEIVE command enables you to read one record from a FIFO, and to display it on the console. The command format is:

```
RECEIVE file
```

where "file" specifies the FIFO you wish to read. This command may be useful where FIFOs are used as message mailboxes in a multi-user system.

Example:

```
0A}RECEIVE MAILBOX.RMN
DICK....NEED TO MEET WITH YOU....JEFF
0A}
```

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

RENAME Command

The RENAME command enables you to rename individual disk files or groups of files. The command format is:

RENAME oldfile newfile ;option

where "oldfile" and "newfile" specify the old and new file names. Wild-card characters ("?" and "*") may be used in the "oldfile" argument to indicate that multiple files are to be renamed. Wild-card characters may also be used in the "newfile" argument to indicate that the corresponding characters of each old file name are to be used in naming each new file. An "oldfile" or "newfile" argument consisting only of a drive letter (e.g., "C:") implies a file specification of all wild cards (i.e., "C:*.*").

If both "oldfile" and "newfile" are omitted from the command line, then the RENAME 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:

oldfile newfile ;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 renamed. If "option" is omitted and "oldfile" contains wild cards, then the RENAME command prompts you to specify whether or not confirmation is desired.

Examples:

```
0A}RENAME ;N
* *.BAK *.PRV
A:REFERENC.BAK renamed to A:REFERENC.PRV
A:USERGUID.BAK renamed to A:USERGUID.PRV
* *.TXT *.BAK
A:LETTER .TXT renamed to A:LETTER .BAK
A:REFERENC.TXT renamed to A:REFERENC.BAK
A:USERGUID.TXT renamed to A:USERGUID.BAK
*
0A}

0A}RENAME B:MAXI*.* OPT*.* ;Y
OK to rename B:MAXICOMP.TXT to B:OPTICOMP.TXT (y/n)? N
OK to rename B:MAXIMUMS.COM to B:OPTIMUMS.COM (y/n)? Y
B:MAXIMUMS.COM renamed to B:OPTIMUMS.COM
OK to rename B:MAXIMUMS.FOR to B:OPTIMUMS.FOR (y/n)? Y
B:MAXIMUMS.FOR renamed to B:OPTIMUMS.FOR
OK to rename B:MAXIMUMS.REL to B:OPTIMUMS.REL (y/n)? N
0A}
```

SEND Command

The SEND command enables you to write one record to a FIFO. The command format is:

SEND file message

where "file" specifies the FIFO you wish to write, and "message" is any text you wish. This command may be useful where FIFOs are used as batch queues or message mailboxes in a multi-user system.

Examples:

```
0A}SEND BATCH.DO DO COMPLINK BIGPROG
0A}SEND MAILBOX.RMN DICK....NEED TO MEET WITH YOU....JEFF
0A}
```

SET Command

The SET command enables you to set and clear the various file attributes, and has the following format:

SET file ;options +onattributes -offattributes

where "onattributes" and "offattributes" consist of any set of the mnemonic letters F, R, G, and A (corresponding to the attributes FIFO, Read-only, Global and Archived). Attributes following the "+" are set, attributes following the "-" are cleared, and all other attributes are left unchanged.

If "file" contains wild-cards, the SET command can modify the attributes of multiple files; the usual "Y" or "N" options may be used to specify whether or not you want to confirm individual files. If "file" is omitted from the command line, then the SET command operates in interactive mode.

Example:

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

sets all .COM files on the default drive to Read-only, Global, and not Archived.

If "file" contains only a drive specification, then the SET command may be used to set a disk volume read-only or read/write, provided no files are open on that drive.

Example:

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

SHOW Command

The SHOW command enables you to display the settings of file attributes. The command format is:

SHOW file ;options

If "file" contains wild-cards, the SHOW command can display the attributes of multiple files. If "file" is omitted from the command line, then the SET command operates in interactive mode. "Options" may include ;Y or ;N to specify whether or not to pause after displaying each file, or ;L to print rather than display.

For example:

```
0A}SHOW B:*.* ;L
```

prints a listing of file attributes for all files on the B-drive.

If "file" contains only a drive specification, then the SHOW command may be used to determine whether a disk volume is read-only or read/write.

Example:

```
0A}SHOW A:
Drive A set to read-only
0A}SHOW B:
Drive B set to read/write
0A}
```

TYPE Command

The TYPE command enables you to display the contents of an ASCII file on the console or printer. The command format is:

```
TYPE file ;L
```

where "file" specifies the disk file to be displayed. If the ";L" suffix is present, then the contents of the file is printed. Otherwise, it is displayed on the console.

Example:

```
0A}TYPE B:TURBODOS.DOC
...ASCII display...
0A}
```

USER Command

The USER command enables you to change the current user number. This command is honored only if you have logged-on as a privileged user. The command format is:

```
USER usernumber
```

where "usernumber" specifies the desired user number (between 0 and 31). If the "usernumber" argument is omitted, then the current user number is simply displayed (it also appears in the command prompt).

Example:

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

VERIFY Command

The VERIFY command enables you to scan a volume (typically a hard-disk) for bad blocks and to mark them so that TurboDOS will avoid their use. The command format is:

VERIFY drive

where "drive" specifies the disk to be verified. If bad blocks are detected, the command will create a read-only directory entry with the filename BLOCKS.BAD to mark the defective blocks.

Use of the VERIFY command is restricted to privileged log-ons only. In a network configuration, VERIFY may not be executed in one processor to access a drive attached to another processor. However, a slave console may be attached to the master processor (see MASTER command) for the purpose of running VERIFY.

(Intentionally left blank)

CP/M- AND MP/M-COMPATIBLE FUNCTIONS

All TurboDOS functions are invoked by means of a call to 0005H with a function number in the C-register. Invalid function numbers result in no operation.

Function 0: Return to Operating System

Called with: C = 0

Returns with: (does not return)

Notes: This function (also called "warm start") is used to terminate execution of a transient program. The same function is more commonly performed by executing a jump to location 0000H, which has exactly the same effect.

Function 1: Input Character from Console

Called with: C = 1

Returns with: A = input character

Notes: The console input character is echoed to console output, with tabs expanded.

Function 2: Output Character to Console

Called with: C = 2
E = output character

Notes: Tabs are expanded.

Function 3: Raw Console Input

Called with: C = 3

Returns with: A = input character

Notes: Input is not echoed to console output. This function is compatible with MP/M-II. (In CP/M, this function is "Input from Reader Device".)

Function 4: Raw Console Output

Called with: C = 4
E = output character

Notes: Tabs are not expanded. This function is compatible with MP/M-II. (In CP/M, this function is "Output to Punch Device".)

Function 5: Output Character to Printer

Called with: C = 5
E = output character

Notes: If the spooler is activated, the character is actually output to a print file.

Function 6: Direct Console Input/Output

Called with: C = 6
E = -1 (for combined console status/input)
-2 (for console status)
-3 (for console input)
output character (for console output)

Returns with: A = input character or status

Notes: If the E-register contains -1, then any available console input character is returned in the A-register (without echo), or if no console input character is available then 0 is returned in the A-register. If the E-register contains -2, then this function is equivalent to function 11 (Return Console Status). If the E-register contains -3, then this function is equivalent to function 3 (Raw Console Input). If the E-register contains a value other than -1, -2 or -3, then this function is equivalent to function 4 (Raw Console Output). This function is MP/M-II compatible (CP/M does not support E=-2 or E=-3).

Function 7: Return I/O Byte

Called with: C = 7

Returns with: A = I/O Byte

Notes: This function simply returns the value of RAM location 0003H.

Function 8: Set I/O Byte

Called with: C =
E = I/O Byte

Notes: This function simply sets the value of RAM location 0003H. It has no effect whatever on TurboDOS I/O assignment.

Function 9: Output Buffer to Console

Called with: C = 9
DE = buffer address

Notes: A string of characters terminated by "$" is output to the console.

Function 10: Input Buffer from Console

Called with: C = 10
DE = buffer address

Notes: The first byte of the buffer must be preset to the maximum number of characters to be input. Console input is accepted until terminated by a carriage-return. Input errors may be corrected by using backspace or delete characters to erase one character, and CTRL-U or CTRL-X to erase the entire line. On return, the second byte of the buffer contains the actual number of characters input from the console. The input string is returned starting in the third byte of the buffer. This function does not expand tabs in TurboDOS (as it does in CP/M and MP/M).

Function 11: Return Console Status

Called with: C = 11

Returns with: A = -1 if console input is available
0 if console input not available

Function 12: Return BDOS Version

Called with: C = 12

Returns with: H = 0 (CP/M, not MP/M)
L = 30H (BDOS Version 3.0)
B = 0 (CP/M, not MP/M)
A = 30H (BDOS Version 3.0)

Notes: The latest compatible CP/M BDOS version number is returned in the L- and A-registers. (Version number returned may be changed by patching symbol CPMVER during system generation.)

Function 13: Reset Disk System

Called with: C = 13

Returns with: A = 0

Notes: In TurboDOS, the only effect of this function is to reset the record buffer address to 0080H.

Function 14: Select Default Drive

Called with: C = 14
E = default drive (0="A", 1="B",..., 15="P")

Function 15: Open File

Called with: C = 15
DE = FCB address

Returns with: A = 0 if successful
-1 if file not found

Notes: The open mode is determined by the "permissive" compatibility flag and the FCB interface attributes f5 and f6, as follows:

Permissive Flag	f6	f5	Open Mode
0	0	0	exclusive
0	0	1	shared
0	1	0	read-only
0	1	1	permissive
1	0	0	permissive
1	0	1	shared
1	1	0	read-only
1	1	1	exclusive

Function 16: Close File

Called with: C = 16
DE = FCB address

Returns with: A = 0 if successful
-1 if file not found

Notes: If FCB interface attribute *f5* is set, then a "partial close" operation is performed which updates the file's directory entry but leaves the file open.

Function 17: Search for First File

Called with: C = 17
DE = FCB address

Returns with: A = 0, 1, 2 or 3 if successful
-1 if file not found

Notes: Returns with directory record (four entries) in record buffer. A-register specifies which of the four directory entries was found.

Function 18: Search for Next File

Called with: C = 18

Returns with: A = 0, 1, 2 or 3 if successful
-1 if file not found

Notes: Continues search initiated by function 17. Continues from last directory entry found. Returns with directory record (four entries) in record buffer. A-register specifies which of the four directory entries was found.

Function 19: Delete File

Called with: C = 19
DE = FCB address

Returns with: A = 0 if successful
-1 if no file found

Notes: If FCB contains wild-cards, all matching files are deleted. If interface attribute f5 is set in the FCB, then the delete function is ignored. A program may delete a file that it has open (the close operation is implied). However, a program is not permitted to delete a file that another process has open, nor a file that has the read-only attribute.

Function 20: Read Next Record

Called with: C = 20
DE = FCB address

Returns with: A = 0 if successful
1 if at end-of-file
2 if reading unwritten data

Function 21: Write Next Record

Called with: C = 21
DE = FCB address

Returns with: A = 0 if successful
1 if file too large (>134 MB)
2 if disk full or write-protected
8 if writing locked record
-1 if no directory space

Function 22: Create File

Called with: C = 22
DE = FCB address

Returns with: A = 0 if successful
-1 if directory full, FCB invalid, or file exists

Notes: The file name in the FCB may not contain wild cards. If FCB interface attribute f5 is set, then the create function leaves the newly-created file open in shared mode. If f5 is not set, then the file is left in either exclusive or permissive mode, depending on the setting of the "permissive" compatibility flag. Attribute f6 is ignored. A request to create a file that already exists is denied.

Function 23: Rename File

Called with: C = 23
DE = FCB address

Returns with: A = 0 if successful
-1 if file not found, in-use, or name invalid

Notes: The first 16 bytes of the FCB contains the file reference to be renamed, and the next 16 bytes contains the new file reference to be used. Wild-cards are not allowed in either file reference. A program may rename a file that it has open (the close operation is implied). However, a program is not permitted to rename a file that another process has open, nor a file that has the read-only attribute.

Function 24: Return Login Vector

Called with: C = 24

Returns with: HL = ready vector

Notes: The login vector contains a one-bit for each drive that is ready, and a zero-bit for each drive that is not ready. The least significant bit corresponds to drive "A", and the most significant bit to drive "P".

Function 25: Return Default Drive

Called with: C = 25

Returns with: A = default drive (0="A", 1="B",..., 15="P")

Function 26: Set Record Buffer Address

Called with: C = 26
DE = record buffer address

Notes: The record buffer address (sometimes called the "DMA address") is set to 0080H by default, unless this function is used to change it to another value.

Function 27: (Return Allocation Vector Address)

Called with: C = 27

Returns with: HL = 0
BA = 0

Notes: Under TurboDOS, this function performs no operation.

Function 28: Write Protect Drive

Called with: C = 28

Notes: Write protects the default drive.

Function 29: Return Write Protect Vector

Called with: C = 29

Returns with: HL = write-protect vector

Notes: The write-protect vector contains a one-bit for each drive that is write-protected, and a zero-bit for each drive that is not. The least significant bit corresponds to drive "A", and the most significant bit to drive "P".

Function 30: Set File Attributes

Called with: C = 30
DE = FCB address

Returns with: A = 0 if successful
-1 if file not found or in-use

Notes: This function searches the directory for the file referenced by the FCB, and updates the file attributes in the directory from those in the FCB. The file attributes are stored in bit 7 (sign bit) of FCB bytes 2 through 12. A program may set attributes on a file that it has open (the close operation is implied). However, a program is not permitted to set attributes on a file that another process has open.

Function 31: Return Disk Parameter Block Address

Called with: C = 31

Returns with: HL = address of DPB

Notes: This function causes TurboDOS to construct a CP/M-style DPB for the default drive.

Function 32: Set/Return User Number

Called with: C = 32
E = user number 0...31 (or -1)

Returns with: A = user number 0...31

Notes: If the E-register contains the value -1, then the current user number is returned in the A-register. If the E-register contains some other value and the user is privileged, then the the current user number is changed to that value (modulo 32).

Function 33: Read Random Record

Called with: C = 33
DE = FCB address

Returns with: A = 0 if successful
1 if reading unwritten data
3 if error changing extents
4 if reading unwritten extent
6 if random record number too large

Notes: This function reads the record number specified by the random record number field (FCB bytes 34 through 36).

Function 34: Write Random Record

Called with: C = 34
DE = FCB address

Returns with: A = 0 if successful
2 if disk full or write-protected
3 if error changing extents
5 if no directory space
6 if random record number too large
8 if writing locked record

Notes: This function writes the record number specified by the random record number field (FCB bytes 34 through 36).

Function 35: Compute File Size

Called with: C = 35
DE = FCB address

Returns with: A = 0 if successful
-1 if file not found

Notes: This function computes the size of the file referenced by the FCB, and returns the number of records in the random record number field (FCB bytes 34 through 36).

Function 36: Set Random Record

Called with: C = 36
DE = FCB address

Notes: This function returns the current file position in the random record number field (FCB bytes 34 through 36). Since the sequential access functions (20 and 21) do not update the random record number field, function 36 should be used when switching from sequential to random access.

Function 37: Reset Write-Protect Vector

Called with: C = 37
DE = reset vector

Notes: This function write-enables the drives which correspond to one-bits in the reset vector. The least significant bit corresponds to drive "A", and the most significant bit to drive "P".

Function 38: (Access Drive)

Called with: C = 38

Returns with: HL = 0
BA = 0

Notes: Under TurboDOS, this function performs no operation.

Function 39: (Free Drive)

Called with: C = 39

Returns with: HL = 0
BA = 0

Notes: Under TurboDOS, this function performs no operation.

Function 40: Write Random Record with Zero Fill

Notes: TurboDOS treats function 40 as a synonym for function 34.

Function 41: (Test and Write Random)

Called with: C = 41

Returns with: HL = 0
BA = 0

Notes: Under TurboDOS, this function performs no operation.

Function 42: Lock Random Record

Called with: C = 42
DE = FCB address

Returns with: A = 0 if successful
1 if positioning to unwritten data
3 if error changing extents
4 if positioning to unwritten extent
6 if random record number too large
8 if record locked by another process

Notes: If the file referenced by the FCB is open in shared mode, then this function attempts to obtain a lock on the specified random record number. If the specified record number is already locked by another process, then this function either suspends or returns an error (A=8) depending on the setting of the "suspend" compatibility flag. If the "logical" compatibility flag is not set, then this function positions the file to the specified record number. TurboDOS sets no limit on how many records a process may have locked at one time. If the random record number field of the FCB is set to the 24-bit value 0FFFFFFH, then this function attempts to obtain an all-inclusive lock (on all records of the file at once). If the referenced file is not open in shared mode, then this function performs no operation and returns a successful result.

Function 43: Unlock Random Record

Called with: C = 43
DE = FCB address

Returns with: A = 0 if successful
1 if positioning to unwritten data
3 if error changing extents
4 if positioning to unwritten extent
6 if random record number too large

Notes: If the file referenced by the FCB is open in shared mode, then this function unlocks the specified random record number. If the "logical" compatibility flag is not set, then this function positions the file to the specified record number. Attempting to unlock a record which was not previously locked does not return an error. If the random record number field of the FCB is set to the 24-bit value 0FFFFFFH, then this function releases any all-inclusive lock, but not any individual record locks. If the referenced file is not open in shared mode, then this function performs no operation and returns a successful result.

Function 44: (Set Multi-Sector Count)

Called with: C = 44

Returns with: HL = 0
BA = 0

Notes: Under TurboDOS, this function performs no operation.

Function 45: (Set BDOS Error Mode)

Called with: C = 45

Returns with: HL = 0
BA = 0

Notes: Under TurboDOS, this function performs no operation.

Function 46: Get Disk Free Space

Called with: C = 46
E = drive (0="A", 1="B",..., 15="P")

Returns with: A = 0

Notes: This function determines the amount of free space on the specified drive. It returns a 24-bit binary value (the number of free 128-byte records) as the first three bytes at the current record buffer (DMA) address, least significant byte first.

Function 47: Chain to Program

Called with: C = 47

Returns with: (does not return)

Notes: A valid TurboDOS command line, terminated by a null byte, must be placed in the default record buffer at address 0080H. This function terminates the invoking program, and then executes the command line.

Function 48: (Flush Buffers)

Called with: C = 48

Returns with: HL = 0
BA = 0

Notes: Under TurboDOS, this function performs no operation.

Function 100: (Set Directory Label)

Called with: C = 100

Returns with: HL = 0
BA = 0

Notes: Under TurboDOS, this function performs no operation.

Function 101: (Return Directory Label Data)

Called with: C = 101

Returns with: HL = .0
BA = 0

Notes: Under TurboDOS, this function performs no operation.

Function 102: (Read File XFCB)

Called with: C = 102

Returns with: HL = 0
BA = 0

Notes: Under TurboDOS, this function performs no operation.

Function 103: (Write File XFCB)

Called with: C = 103

Returns with: HL = 0
BA = 0

Notes: Under TurboDOS, this function performs no operation.

Function 104: Set Date and Time

Called with: C = 104
DE = Address of 4-byte date/time packet

Notes: Registers DE point to a 4-byte date/time packet with the following structure:

```
.WORD   Binary Julian date with zero -> 31 December 1977
.BYTE   Hours represented as two BCD digits
.BYTE   Minutes represented as two BCD digits
```

Seconds are set to zero.

Function 105: Return Date and Time

Called with: C = 105
DE = Address of 4-byte date/time packet

Notes: Registers DE point to a 4-byte area in which a date/time packet is returned. The structure of this packet is the same as for function 104.

Function 106: (Set Default Password)

Called with: C = 106

Returns with: HL = 0
BA = 0

Notes: Under TurboDOS, this function performs no operation.

Function 107: Return CP/M Serial Number

Called with: C = 107
DE = Address of 6-byte serial-number packet

Notes: Under TurboDOS, this function always returns six zero bytes.

File Control Block Organization

The File Control Block (FCB) consists of an area of 33 bytes for sequential file access, or 36 bytes for random file access. The FCB organization is shown below:

Bytes	Field	Description
1	Drive	0=default, 1="A", 2="B",..., 16="P"
2...9	Name	file name in ASCII, right-padded with spaces sign bits reserved for file attributes
10...12	Type	file type in ASCII, right-padded with spaces sign bits reserved for file attributes
13	Extent	least significant 5 bits of extent number
14	Flags	used by TurboDOS (Do Not Use)
15	Extent'	most significant 8 bits of extent number
16	Record Count	number of records in current extent 0...128 (Do Not Use)
17-32	Map	allocation map of current extent (Do Not Use)
33	Current Record	current record number 0...127 in current extent used and incremented by sequential read and write
34-36	Random Record	20-bit record number used by random read, write, lock and unlock (byte 34 is least significant)

Simulated CP/M BIOS Branch Table

TurboDOS 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 255 bytes below the top of memory in order to ensure that it begins on a page boundary (as it does in CP/M). The entry points are:

```
xF00H   JMP     BOOT    ; Cold Start
xF03H   JMP     WBOOT   ; Warm Start
xF06H   JMP     CONST   ; Console Status to A-reg
xF09H   JMP     CONIN   ; Console Input to A-reg
xF0CH   JMP     CONOUT  ; Console Output from C-reg
xF0FH   JMP     LIST    ; Printer Output from C-reg
xF12H   JMP     RAWOUT  ; Raw Console Output from C-reg
xF15H   JMP     RAWIN   ; Raw Console Input to A-reg
xF18H   JMP     HOME    ; Set Track to Zero
xF1BH   JMP     SELDSK  ; Select Disk Drive from C-reg
xF1EH   JMP     SETTRK  ; Set Track from BC-regs
xF21H   JMP     SETSEC  ; Set Sector from BC-regs
xF24H   JMP     SETDMA  ; Set DMA Address from BC-regs
xF27H   JMP     READ    ; Read Disk Sector
xF2AH   JMP     WRITE   ; Write Disk Sector
xF2DH   JMP     LISTST  ; List Status to A-reg
xF30H   JMP     SECTRN  ; Sector Translate Subroutine
```

Since reader and punch devices are not implemented in TurboDOS (as in MP/M-II), the branch table entries corresponding to CP/M reader and punch cause raw console input and output instead. The disk read and write entrypoints are honored only for privileged users.

(Intentionally left blank)

ADDITIONAL TURBODOS FUNCTIONS

Additional TurboDOS functions use function numbers in the ranges 64...99 and 108...127, and are intended primarily for use by TurboDOS commands and internal processes. These functions may be used safely by user programs, except for those marked "Do Not Use" and those with cautionary notes.

Function 74: Return Disk Allocation Information

Called with: C = 74
E = drive (0="A", 1="B",..., 15="P")

Returns with: A = block size (3=1K, 4=2K,..., 7=16K)
C = directory blocks
DE = free-space blocks
HL = total blocks

Function 75: Return Physical Disk Information

Called with: C = 75
E = drive (0="A", 1="B",..., 15="P")

Returns with: A = sector size (0=128, 1=256, 2=512,..., 7=16K)
BC = reserved tracks
DE = tracks per disk
HL = sectors per track

Function 76: Set/Return Print Mode

Called with: C = 76
E = print mode:
0 to print direct
1 to print spooled
2 to print to the console
-1 to leave print mode unchanged
D = printer assignment (if print mode = 0)
queue assignment (if print mode = 1)
-1 to leave assignment unchanged
B = spool drive (0="A", 1="B",..., 15="P")
-1 to leave spool drive unchanged

Returns with: A = current spool drive (0="A", 1="B",..., 15="P")
H = current printer or queue assignment
L = current print mode

Notes: Printer and queue assignments are coded 1="A", 2="B",..., 16="P". Assignment to queue zero causes print files to be left unqueued. Assignment to printer zero causes print output to be discarded. Setting the assignment, mode, or spool drive implies an immediate end-of-print-job condition. If both B = D = E = -1, then this function returns the current assignment, mode, and drive, and does not signal end-of-print-job.

Function 77: Signal End-of-Print-Job

Called with: C = 77

Notes: This function signals an end-of-print-job condition. If the current print mode is spooling, then the current print file (if any) is closed and (if appropriate) queued for de-spooled printing.

Function 78: Set/Return De-Spool Mode

Called with:
- C = 78
- B = printer (0="A", 1="B",..., 15="P")
- E = de-spool mode:
  - 0 to process print job
  - 1 to suspend print job
  - 2 to restart print job from the beginning
  - 3 to terminate print job
  - -1 to leave de-spool mode unchanged
- D = de-spool queue assignment:
  - 1="A", 2="B",..., 16="P"
  - 0 to set printer offline
  - -1 to leave queue assignment unchanged

Returns with:
- A = 0 if successful
  - -1 if invalid
- H = current queue assignment 0...16
- L = current de-spool mode (1 if stopped, 0 otherwise)

Notes: If both D- and E-registers are -1, then this function returns the current queue assignment and de-spool mode for the specified printer.

Function 79: Queue a Print File

Called with:
- C = 79
- DE = FCB address
- H = queue (0="A", 1="B",..., 15="P")
- L = user number (0...31) plus
  - bit 7 = 1 if queued files to be deleted after printing

Returns with:
- A = 0 if successful
  - -1 if invalid

Notes: Only the first 16 bytes of the FCB are used. The specified FCB-drive must be accessible by the processor in which the specified queue resides, otherwise the request is invalid. If this function is called with L=-1, then the validity check of the FCB-drive and queue are performed, but no file is queued.

Function 80: (Reserved)

Notes: Do Not Use.

Function 81: (Reserved)

Notes: Do Not Use.

Function 82: (Reserved)

Notes: Do Not Use.

Function 83: Set Date & Time

Called with: C = 83
HL = Julian date
D = hours (0...23)
E = minutes (0...59)
B = seconds (0...59)

Notes: The Julian date in HL should be the number of days since the base date of December 31, 1947. Dates prior to the base date are represented by negative numbers.

Function 84: Return Date & Time

Called with: C = 84

Returns with: HL = Julian date
D = hours (0...23)
E = minutes (0...59)
B = seconds (0...59)

Notes: The Julian date in HL is be the number of days since the base date of December 31, 1947. Dates prior to the base date are represented by negative numbers.

Function 85: Set/Return Drive Status

Called with: C = 85
E = drive (0="A", 1="B",..., 15="P")
D = operation code:
0 to set the drive read/write
1 to set the drive read-only
-1 to return the current drive status

Returns with: A = 0 if successful
-1 if attempt to set drive status while files open
L = drive ready status:
0 if drive is not ready
-1 if drive is ready
H = drive read-only status:
0 if drive is read/write
-1 if drive is read-only

Function 86: Physical Disk Access

Called with: C = 86
DE = PDR packet address

Returns with: A = 0 or -1

Notes: This function is available for privileged log-ons only. It provides direct access to the physical disk drivers, in order to support the BACKUP and FORMAT commands. On entry, the DE-registers contain the address of a 14-byte Physical Disk Request (PDR) packet. The first byte of the PDR packet is an operation code which determines the physical operation to be performed. Refer to the disk driver interface specification in the Configuration Guide for details.

Function 87: Return Comm Channel Status

Called with: C = 87
D = channel number, plus
bit 7 = 1 if remote channel via network

Returns with: A = 0 if comm channel input not available
A = -1 if comm channel input is available

Function 88: Input Character from Comm Channel

Called with: C = 88
D = channel number, plus
bit 7 = 1 if remote channel via network

Returns with: A = input character

Function 89: Output Character to Comm Channel

Called with: C = 89
D = channel number, plus
bit 7 = 1 if remote channel via network
E = output character

Function 90: Set Comm Channel Baud Rate

Called with: C = 90
D = channel number, plus
bit 7 = 1 if remote channel via network
E = baud rate code:
bit 7 = 1 if attention detection enabled
bit 6 = 1 if clear-to-send handshaking enabled
bit 5 = 1 if output-only (input disabled)
bits 3-0 = baud-rate value 0...15 (see table below)

Notes: The least significant nibble of the E-register contains a baud rate value as follows:

0 =	50	8 =	1,800
1 =	75	9 =	2,000
2 =	110	10 =	2,400
3 =	134.5	11 =	3,600
4 =	150	12 =	4,800
5 =	300	13 =	7,200
6 =	600	14 =	9,600
7 =	1,200	15 =	19,200

Function 91: Return Comm Channel Baud Rate

Called with: C = 91
D = channel number, plus
bit 7 = 1 if remote channel via network

Returns with: A = baud rate code

Notes: A-register returns a baud rate code as described for function 90.

Function 92: Set Modem Controls

Called with: C = 92
D = channel number, plus
bit 7 = 1 if remote channel via network
E = modem control vector:
bit 7 = request-to-send
bit 6 = data-terminal-ready

Function 93: Return Modem Status

Called with: C = 93
D = channel number, plus
bit 7 = 1 if remote channel via network

Returns with: A = modem status vector:
bit 7 = clear-to-send
bit 6 = data-set-ready
bit 5 = data-carrier-detect
bit 4 = ring-indicator

Function 94: (Reserved)

Notes: Do Not Use.

Function 95: (Reserved)

Notes: Do Not Use.

Function 96: Set Buffer Parameters

Called with: C = 96
D = number of buffers
E = buffer size (0=128, 1=256, 2=512,.., 7=16K)

Notes: Number of buffers must be at least 2. Buffer size must be at least as large as the largest physical disk sector size in use. This function causes all buffers to be written out (if necessary) and placed on the free chain. If this function is performed in a processor without local disk storage, then the function is passed over the network.

Function 97: Return Buffer Parameters

Called with: C = 97

Returns with: A = memory size (pages, 0=64K)
H = number of buffers
L = buffer size (0=128, 1=256, 2=512,.., 7=16K)

Notes: If this function is performed in a processor without local disk storage, then the function is passed over the network.

Function 98: Activate DO-File

Called with: C = 98
DE = FCB address (or 0)

Returns with: A = 0 if successful
-1 if file not found

Notes: Only the first 16 bytes of the FCB are used. The file need not have been previously opened. This function causes any currently-active DO-file and command line to be stacked, and a new DO-file to be activated. This function may be called with DE=0 to cancel all active and stacked DO-files.

Function 99: Disable/Enable Autoload

Called with: C = 99
E = 0 to disable autoload
-1 to enable autoload

(Functions 100...107 described in Section 4)

Function 108: Send Command Line

Called with: C = 108
DE = buffer address (or 0)

Notes: The first byte of the buffer must contain the command line length, with the command line text starting at the second byte of the buffer. This function causes any currently-active command line to be stacked, and the new command line to be activated. This function may be called with DE=0 to cancel all active

and stacked command lines.

Function 109: Set Error Intercept Address

Called with: C = 109
DE = address of error intercept routine (or 0)

Notes: This function allows a program to establish its own error intercept routine to intercept unrecoverable disk errors. Normal TurboDOS error diagnosis is supressed. The error intercept routine must not make any operating system calls, and must return with the A-register set to the desired error recovery alternative (A=0 to retry, A=+1 to ignore, A=-1 to abort). If this function is called with DE-registers set to zero, then normal TurboDOS error diagnosis is restored.

Function 110: Set Abort Intercept Address

Called with: C = 110
DE = address of abort intercept routine (or 0)

Notes: This function allows a program to establish its own abort intercept routine to intercept user-requested aborts (in response to an attention-request or a disk error). The abort intercept routine may exit via a RET instruction to resume execution of the program at the point of interruption. If this function is called with DE-registers set to zero, then normal TurboDOS abort processing is restored.

Function 111: Log-On/Log-Off

Called with: C = 111
E = user number 0...31 to log-on
with bit 7 = 1 for privileged
-1 to log-off

Returns with: A = 0 if successful
-1 if unsuccessful

Function 112: Lock/Unlock Drive

Called with: C = 112
E = drive (0="A", 1="B",..., 15="P")
D = 0 to unlock drive
-1 to lock drive

Returns with: A = 0 if successful
-1 if unsuccessful

Notes: This function attempts to lock the specified drive. If the specified drive is in use or already locked by another user, then the lock request is denied.

Function 113: Load Program

Called with: C = 113
DE = FCB address

Returns with: A = 0 if successful
1 if not enough memory to load file
-1 if file not found

Notes: The file need not have been previously opened. This function loads the program file specified by the FCB into memory starting at the current record buffer address. It makes use of the TurboDOS program load optimizer if available.

Function 114: Rebuild Disk Allocation Map

Called with: C = 114
E = drive (0="A", 1="B",..., 15="P")

Returns with: A = 0 if successful
-1 if disk is write-protected or has files open

Function 115: Flush/Free Buffers

Called with: C = 107
E = drive (0="A", 1="B",..., 15="P")
D = 0 to flush but not free buffers
-1 to flush and free buffers

Notes: This function writes out to disk any buffers belonging to the specified drive which have been written to. If D=-1, it then places on the free chain all buffers belonging to the specified drive.

Function 116: (Reserved)

Notes: Do Not Use.

Function 117: (Reserved)

Notes: Do Not Use.

Function 118: Remote Console Input/Output

Called with: C = 118
E = console input character (or 0)
D = 0 to detach remote console
-1 to attach remote console

Returns with: A = 0 if CONREM is not present
+1 if successful
-1 if executing in master

Notes: This function works in conjunction with the CONREM console driver module to support the MASTER command. It returns a count byte and up to 127 bytes of console output at the current record buffer address.

Function 119: Return TurboDOS Serial Number

Called with: C = 119

Returns with: HL = TurboDOS Origin Number
DE = TurboDOS Unit Number
B = 0 if non-privileged log-on
80H if privileged log-on
C = 12H (TurboDOS Version 1.2)

Function 120: Set Compatibility Flags

Called with: C = 120
E = compatibility flags:
bit 7 = Permissive
bit 6 = Suspend
bit 5 = Global Write
bit 4 = Mixed Mode
bit 3 = Logical
bits 2-0 = (not defined)

Notes: The compatibility flags affect the rules by which file sharing is done. They have no effect in single-user TurboDOS systems. The meaning of each flag was described in Section 2.

Function 121: Allocate Memory Segment

Called with: C = 121
DE = length of segment requested

Returns with: HL = address of segment
A = 0 if successful
-1 if insufficient memory

Notes: Not intended for use by transient programs, and must be used with great care. If allocated segment is not deallocated before program terminates, then the space is permanently lost.

Function 122: De-Allocate Memory Segment

Called with: C = 122
DE = address of segment

Notes: Not intended for use by transient programs, and must be used with great care. If the address passed in DE is not the address of a segment allocated by function 121, then TurboDOS may crash.

Function 123: Send Inter-Process Message

Called with: C = 123
DE = address of message node
HL = address of message

Notes: Not intended for use by transient programs, and must be used with great care. A message node is a 12-byte structure which must be initialized as follows:

```
.WORD  0
.WORD  .
.WORD  .-2
.WORD  .
.WORD  .-2
```

Function 124: Receive Inter-Process Message

Called with: C = 124
DE = address of message node

Returns with: HL = address of message

Notes: Not intended for use by transient programs, and must be used with great care.

Function 125: Delay Process

Called with: C = 125
DE = delay count (system ticks) or 0

Notes: If the delay count is nonzero, then the calling program is suspended for the requested number of system ticks. A system tick is an implementation-dependent time interval, usually about 1/60 of a second. If the delay count is zero, then the calling program is suspended only long enough to allow any other ready processes to run (a "courtesy" dispatch).

Function 126: Create Process

Called with: C = 126
DE = address of process entrypoint (or 0)
HL = address of process work area

Returns with: A = 0 if successful
-1 if insufficient memory

Notes: Not intended for use by transient programs, and must be used with great care. If registers DE are nonzero, this function creates a new process which starts execution at the entrypoint specified in the DE-register. The new process is automatically assigned a TurboDOS work area whose address appears to the new process in the X-register, and a 64-word stack area whose address appears in the SP-register. If the process requires a re-entrant work area (usually dynamically allocated), its address should be passed in the HL-register and will appear to the new process in the Y-register.

If registers DE are zero, this function causes the calling process to terminate.

Function 127: User Defined Function

Called with: C = 127
B = network routing:
0 if always processed locally
1d hex if routed to network address of drive "d"
2p hex if routed to network address of printer "p"
3q hex if routed to network address of queue "q"
-1 if routed to default network address DEFDID
DE = user defined parameter passed
HL = user defined parameter passed

Returns with: A = user defined parameter returned
BC = user defined parameter returned
DE = user defined parameter returned
HL = user defined parameter returned

Notes: This function provides a means for adding user-defined extensions to the TurboDOS operating system which may take full advantage of the TurboDOS networking facilities. On call, the B-register defines how the request is to be routed over the network. Registers DE and HL and the 128-byte record buffer (at current DMA address) are all passed (over the network if necessary) to a user-defined module with the public entrypoint symbol USRFCN::. Upon entry to USRFCN::, the BC-registers contain the address of the 128-byte record that was passed. USRFCN:: may return information to the caller in all of the seven registers ABCDEHL and in the record buffer.

APPENDIX — APPLICATION NOTES

TurboDOS Application Note

Digital Research PIP Utility

The use of the Digital Research utility PIP is not recommended under TurboDOS, since PIP is confused by file attributes and does not preserve them when copying. PIP also destroys the flag byte (byte 14) of FCBs, which is a "no-no" under TurboDOS. The TurboDOS COPY command does not have these problems.

If PIP must be used under TurboDOS, the following patch should be made to PIP.COM to alleviate the flag-byte problem:

At 1939 change D2 to C3
At 193A change 45 to 4A
At 1958 change 34 to 00

This patch is for PIP Version 1.5, so you should verify that you have this version by DUMPing PIP.COM and looking at the version identification which follows the copyright notice approximately 256 bytes into the .COM file.

The TurboDOS MONITOR command is ideal for making this patch. Use the "L" directive to load PIP.COM into RAM, use the "E" directive to make the changes, and use the "S" directive to save the patched version in a new .COM file.

TurboDOS Application Note

Digital Research ED Line Editor

The Digital Research line editor ED destroys the flag byte (byte 14) of FCBs, which is a "no-no" under TurboDOS. If ED must be used under TurboDOS, the following patch should be made to ED.COM to alleviate this problem:

At 0FCE change 36 to 00
At 0FDF change 32 to 00
At 0FE0 change 4A to 00
At 0FE1 change 1B to 00

The TurboDOS MONITOR command is ideal for making this patch. Use the "L" directive to load ED.COM into RAM, use the "E" directive to make the changes, and use the "S" directive to save the patched version in a new .COM file.

Application Notes

TurboDOS Application Note

MicroPro WordStar Word Processor

When using MicroPro WordStar under TurboDOS, operation tends to be sluggish, especially during printing. This is due to the fact that WordStar makes very frequent calls to the "console status" function, which involves a great deal more overhead in TurboDOS than it does in CP/M. The following patch to WordStar cuts down on the frequency of "console status" calls, and thereby eliminates the performance problem.

This patch must be applied to the installed version of WordStar WS.COM, not to the uninstalled version WSU.COM. The patch has been exhaustively tested with WordStar 3.0, but will probably work with other versions:

```
At 02AE change 0A to 00    (DELCUS = 0)
At 02AF change 05 to 00    (DELMIS = 0)

At 02BA change 00 to C3    (UCNSTA = JMP to patch area)
At 02BB change 00 to E0
At 02BC change C9 to 02

At 02D1 change 19 to 04    (DEL3 = 4)
At 02D2 change 40 to 08    (DEL4 = 8)

At 02E0 change 00 to 21    (Patch area)
At 02E1 change 00 to EF
At 02E2 change 00 to 02
At 02E3 change 00 to 34
At 02E4 change 00 to CB
At 02E5 change 00 to 6E
At 02E6 change 00 to 3E
At 02E7 change 00 to 00
At 02E8 change 00 to C8
At 02E9 change 00 to 77
At 02EA change 00 to 0E
At 02EB change 00 to 0B
At 02EC change 00 to C3
At 02ED change 00 to 05
At 02EE change 00 to 00
At 02EF change 00 to 00

At 0717 change 00 to 00    (CSWTCH = 0)
At 0718 change 00 to FF    (HAVBSY = FF)
```

The TurboDOS MONITOR command is ideal for making this patch. Use the "L" directive to load WS.COM into RAM, use the "E" directive to make the changes, and use the "S" directive to save the patched version in a new .COM file.

Update U1.62831.1282 (8222 900 50956)

Module F51H "P3500 TURBODOS USER'S GUIDE FOR PROGRAMMERS"

This Update states the TurboDOS 1.22 amendments for module F51H, valid from Release 1.1 onwards.

MANUAL STATUS SURVEY

Module F51H "P3500 TURBODOS USER'S GUIDE FOR PROGRAMMERS"

For this manual the following updates have been issued:

- U1.62831.1282 (8222 900 50956; December 1982 update for release 1.1)

software 2000 inc.

1127 Hetrick Avenue • Arroyo Grande, California 93420

TurboDOS 1.22 Documentation Update

December, 1982

USER'S GUIDE REVISIONS

The following revisions should be made to the User's Guide to TurboDOS 1.2 (May, 1982):

On page 1-2 (in the "Improved Performance" paragraph), add:

> In addition to ordinary CP/M-compatible linear directories, TurboDOS also supports an optional hash-encoded directory format which employs a hashing algorithm to make look-up in large directories much, much faster.

On page 2-4 (after the "Disk Formats" section), add:

Directory Formats

TurboDOS supports two alternate directory formats: linear and hashed. The standard linear directory format is compatible with CP/M, and is searched sequentially. Consequently, directory look-up speed deteriorates with increasing directory size, and can get rather slow on hard disks which contain a large number of directory entries. The linear directory format must be used for disks transported to or from non-TurboDOS systems.

The optional hashed directory format uses a hashing algorithm to make look-up in large directories much, much faster. A hashed directory may be used on any disk, but is especially suited for use on hard disks which contain a large number of directory entries. To set up a disk to use this optional hashed directory format, use the ERASEDIR command and answer "Y" to the prompt "Hashed directory desired (Y/N)?" Whether a directory is to be maintained in hashed or linear format is recorded in the directory label. The DIR command displays "(H)" to indicated that a directory is hashed. Hashed directories are not compatible with CP/M, or with older versions of TurboDOS prior to 1.21.

Note that directory searches which involve "wild cards" have to be done linearly regardless of whether the directory format is linear or hashed. In fact, such wild-card searches are typically slower if the directory is hashed.

On page 2-10, delete the final paragraph concerning "SGLLOG".

On page 2-12 (File Attributes), change the paragraph on the "archived" attribute to read:

> 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 or renamed.

On page 2-14 (File-Level Interlocks), add to the paragraph on the "permissive" open-mode:

> The exclusive write-lock is not gained when writing to a FIFO file.

On page 2-17 (File-Sharing Compatibility Modes), add to the paragraph on the "global-write" flag:

> Access to global FIFO files is always read/write regardless of the setting of the "global write" compatibility flag.

On page 2-20 (after the "Disk Error Handling" section), add:

> When spooling to disk, a full-disk condition causes the TurboDOS spooler to close the current print file (prematurely) and not queue it for de-spooled printing. TurboDOS then displays the following diagnostic message:
>
> Spooler Error (Ignore, Abort)
>
> and waits for the operator to choose the desired error recovery option by keying the appropriate letter ("I" or "A"). The "Ignore" response causes the print mode to be set to "offline" and the user's program to continue with further print output discarded. The "Abort" response causes the user's program to be terminated. In either case, the incomplete print file will not be printed unless it is manually queued with a QUEUE command by the operator.

During de-spooled printing from disk, any unrecoverable disk error causes the TurboDOS de-spooler to abort the print job in process and to place the printer in "offline" status. Any remaining print jobs remain queued. No diagnostic message is displayed (since the de-spooler is not attached to any console).

On page 2-21 (after the "Memory Management" section), add:

Sometimes it is desirable to gain additional TPA space by reducing the size of the disk buffer pool. This can be accomplished with the BUFFERS command, provided that there is no other dynamic space allocated below the disk buffer pool. Therefore, it is necessary to make sure that TurboDOS is quiescent (no programs running, no files open, no print jobs queued, etc.) before changing disk buffer parameters. In networking configurations, it is also necessary that TurboDOS is generated with an adequate number of pre-allocated message buffers and reply packets (see Configuration Guide). Otherwise, the BUFFERS command will still cause the disk buffer pool to get smaller, but the TPA will not get correspondingly larger.

On page 3-1 (AUTOLOAD Command), change the second paragraph to read:

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

AUTOLOAD commandstring

where "commandstring" is any valid TurboDOS command or a string of commands separated by a special delimiter, the vertical bar character "|". The AUTOLOAD command converts each vertical bar "|" to a backslant "\" as it is processing the command string.

and at the bottom of page 3-1, add:

NOTE: The command processor AUTOLOAD.COM has two patchable locations. Location 103H contains the special command separator character and defaults to "|". Location 104H contains the substituted command separator character and defaults to "\".

NOTE: In order to work with release 1.22, all .AUT files created with AUTOLOAD under prior releases must be regenerated.

On page 3-3 (BATCH Command), change the "NOTE" to read:

NOTE: The command processor BATCH.COM has three patchable locations. Location 103H contains the special command separator character and defaults to "|". Location 104H contains the substituted command separator character and defaults to "\". Location 105H contains the drive on which the FIFO BATCH.DO is located and defaults to 0 (drive A).

On page 3-13 (DIR Command), add:

The directory display consists of a preamble followed by a list of files. The preamble includes the following information:

- disk label
- date and time
- the symbol "(H)" if the directory is hashed
- free space remaining on the disk
- number of files displayed
- requested drive and file specification
- combined size of files displayed

The list of files is alphebetized, and the size of each files is shown. Read-only files are distinguished by a colon ":" (rather than a period ".") preceding the file type. If there are too many files to fit on the screen, then the DIR command waits for a carriage-return at the end of each screenful.

On page 3-14 (DO Command), add to the end of the the second paragraph:

If the DO command is invoked by a privileged user under user-number zero, and if the original DO-file has the global attribute, then the temporary DO$-file is also given the global attribute.

and at the bottom of page 3-15, add:

NOTE: The command processor DO.COM has two patchable locataions. Location 103H contains the left parameter delimiter and defaults to "{". Location 104H contains the right parameter delimiter and defaults to "}".

On page 3-16 (DRIVE Command), add to the end of the example display:

```
Fixed (or removable) disk
```

On page 3-18 (ERASEDIR Command), change the example to:

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

On page 3-25 (MASTER Command), add:

NOTE: Do not attempt to run the MASTER command from more than one console at a time. If you do, console output from the master processor will be randomly distributed across two or more consoles, and will consequently be undecipherable. If this should ever occur by mistake, simply detatch all but one of the consoles.

NOTE: The command processor MASTER.COM has two patchable locations. Location 103H contains the special attention character and defaults to "^A". Location 104H contains the substituted attention character and defaults to "^S".

On pages 3-26 and 3-27 (MONITOR Command), change the syntax of the Load and Save directives to:

```
L  filename[ addr]
S  filename[ addr1,addr2]
```

On page 3-36 (SEND Command), add to the bottom of the page:

> NOTE: The command processor SEND.COM has two patchable locations. Location 103H contains the special command separator character and defaults to "|". Location 104H contains the substituted command separator character and defaults to "\".

On page 4-1 (introduction to CP/M-Compatible Functions), add:

> TurboDOS function calls generally destroy registers A-B-C-D-E-H-L. CP/M-compatible functions which return an 8-bit value in the A-register also return the same value in the L-register, and set both B- and H-registers to zero.

On page 4-6 (Function 17), change "Notes" to read:

> **Notes:** Returns with a directory record (four entries) in the record buffer, and with the A-register specifying which of the four entries was found. Function 17 searches the disk directory for the first file that matches the specified FCB. The FCB may contain wild cards. If the first byte of the FCB (drive) is set to a "?", then the first directory entry of the default drive is returned unconditionally (in most cases, the label entry).

On page 4-6 (Function 18), change "Notes" to read:

> Returns with a directory record (four entries) in the record buffer, and with the A-register specifying which of the four entries was found. Function 18 continues the search initiated by function 17, continuing from the last directory entry found. If the first byte of the FCB (drive) is set to a "?", then the next directory entry of the default drive is returned unconditionally, regardless of whether it is a valid file entry, a deleted file entry entry, or an allocation map entry.

On page 5-1 (introduction to Additional TurboDOS Functions), add:

> TurboDOS function calls generally destroy registers A-B-C-D-E-H-L.

On page 5-1 (Function 74), change the first item of "Returns with:" to read:

A = block size (3=1K, 4=2K,..., 7=16K), plus bit 7 = 1 if fixed disk

On page 5-12 (Function 115), change the "Called with" section to read:

C = 115
E = drive (0="A", 1="B",..., 15="P")
D = option flags:
bit 7 = 1 to free buffers
bit 6 = 1 to free buffer after disk error "abort" response
bit 5 = 1 to continue after disk error "abort" response
bit 4 = 1 to return after disk error "abort" response

On page 5-14 (Function 123), change "12-byte structure" to "5-word structure".

In Appendix (MicroPro WordStar Applications Note), change the first two bytes of the patch from 00 to 01, as follows:

At 02AE change 0A to 01 (DELCUS = 1)
At 02AF change 05 to 01 (DELMIS = 1)
(...remainder of patch unchanged...)

LIST OF CHANGES IN RELEASE 1.21

1. **Error in Compute File Size (function 35) fixed:**
This error caused all files under user zero to be accessable whether or not the Global attribute was set.

2. **Error in NETMGR module fixed:**
This error caused message buffers to be discarded instead of re-used, resulting in erosion of TPA space, trapping of disk buffers, and possible system crashes. A temporary patch was issued to fix this problem in 1.20.

3. **Error in Load Program (function 113) fixed:**
This error caused program files to be opened in default mode rather than read-only mode. If the default open mode was Exclusive, then all but the first of multiple near-simultaneous program load or auto-load sequences would fail. A temporary patch was issued to fix this problem in 1.20.

4. **The SGLLOG module has been deleted:**
This module can no longer be supported due to the more generalized topology of TurboDOS networks introduced in 1.20.

5. **Error in Log-On/Log-Off (function 111) fixed:**
This error caused all files in user zero with the Global attribute set to be accessible when logged off, which defeated log-on security.

6. **Error in DATE command fixed:**
The sequence ESCAPE-I is no longer output to the console when re-displaying date and time prompts.

7. **Error in LOGON command fixed:**
This error caused LOGON to fail if a SYSLOG.SYS file was present, resulting in endless auto-loading of the LOGON command.

8. **Error in USER command fixed:**
This error caused the "Non-priveleged user" diagnostic to be followed by trailing trash on the console.

9. **Error in DIR command fixed:**
This error caused the three-column display mode (used when files of a megabyte or more are encountered) to exceed an 80-column display width.

10. **Enhancement to GEN command:**
The GEN command now diagnoses and ignores additional defined starting addresses (transfer addresses) after the first one.

11. **Error in MONITOR command fixed:**
This error caused hex digits A-F entered in lower case while in "Examine" mode to be ignored.

12. **Enhancement to network download facility:**
When a slave processor sends a download request message, it is no longer necessary for the message header to have MSGOID set equal to MSGSID by the driver; this is now done by TurboDOS internally. (See Configuration Guide, page 3-19.)

13. **Change to Return Buffer Parameters (function 97):**
The memory size returned in the A-register is now the memory size of the processor where the function originated, rather than the processor where the function was processed.

14. **Enhancement to NETMGR to pre-allocate reply packets:**
A new patchable system parameter, NMBRPS, has been added to the NETMGR module. It may be patched to a positive 8-bit value to pre-allocate reply packets, in the same fashion as NMBMBS pre-allocates message buffers. This may be used to prevent trapping of disk buffers.

15. **Error in VERIFY command fixed:**
This error caused duplicate map entries to be made occasionally in the BLOCKS.BAD file, consuming excess map area in the directory entry.

16. **Error in VERIFY command fixed:**
This error caused insufficient TPA not to be diagnosed properly, generally resulting in a system crash.

17. **User sign-on message capability added:**
An optional user-supplied sign-on message may now be defined. It must begin with the public entry name "USRSOM::" and terminated with a "$". If USRSOM is defined, then it will be displayed immediately following the normal TurboDOS sign-on message.

18. **Error in COPY, DELETE, DIR, QUEUE, RENAME, SERIAL, SET and SHOW commands fixed:**
This error caused insufficient TPA not to be diagnosed properly while building a sorted directory list, generally resulting in a system crash.

19. **Enhancement to COPY, DELETE, DIR, QUEUE, RENAME, SERIAL, SET and SHOW commands:**
The directory sort algorithm has been speeded up. It now sorts 1,000 files twice as fast as before.

20. **Error in MONITOR command fixed:**
This error caused the message "linker control error" to be displayed when attempting to load the MONITOR command into a TPA of insufficient size.

21. **Error in write to un-allocated block fixed:**
This error caused files extended in certain non-sequential orders (very rare) to become corrupted.

22. **Error in record locks fixed:**
This error caused an error-return if an attempt was made to write to a locked record when the Logical compatability flag was set.

23. **Addition of PACKAGE command:**
The PACKAGE command is a librarian utility for concatenating Microsoft-format .REL files. The command syntax for PACKAGE is identical to the GEN command, except that the input filename defaults to type ".PKG" (instead of ".GEN") and the output filename defaults to type ".REL". This command may be used to construct custom packages of TurboDOS modules, make additions to the supplied "STD" packages, pre-package collections of driver modules, etc.

24. **Enhancement to FIFO manager:**
If a RAM FIFO is not currently open by any process but does contain records, those records will be discarded if either (1) the FIFO is renamed, or (2) the FIFO attribute (F1) is turned off.

25. **Major enhancement: hashed directories:**
Disk directories will be maintained in an optional hashed format if so requested during the ERASEDIR command. Whether a directory is maintained as linear or hashed is recorded in the directory label. Hashed directories are <u>not</u> compatible with CP/M or previous TurboDOS versions. The DIR command displays the symbol "(H)" in its preamble to indicate the presence of a hashed directory. Hashing provides a dramatic improvement to non-wild-card directory searches on disks with large directories (e.g., hard disks).

26. **Fixed disk flag added to disk specification table:**
Bit 7 of the first byte of a DST (the block size byte) now signifies that the disk is non-removable, and enables various performance-enhancing shortcuts in TurboDOS which are possible for fixed disks. The DRIVE command now displays whether a disk is fixed or removable.

27. **PATCH module lengthened:**
The PATCH module has been lengthened from 64 bytes to 128 bytes.

28. **Error in OSBOOT fixed:**
This error caused the first OSLOAD.COM entry found in the directory to be loaded, whether or not the user number was zero.

LIST OF CHANGES IN RELEASE 1.22

1. **Error in COPY command fixed:**
This error caused the archive option ";A" to copy all files (including those with the archived attribute).

2. **Enhancement to spooler error handling:**
When spooling to disk, a full-disk condition now causes the TurboDOS spooler to close the current print file (prematurely) and not queue it for de-spooled printing. TurboDOS then displays the following diagnostic message:

    Spooler Error (Ignore, Abort)

    and waits for the operator to choose the desired error recovery option by keying the appropriate letter ("I" or "A"). The "Ignore" response causes the print mode to be set to "offline" and the user's program to continue with further print output discarded. The "Abort" response causes the user's program to be terminated. In either case, the incomplete print file will not be printed unless it is manually queued with a QUEUE command by the operator.

3. **Enhancement to de-spooler error handling:**
During de-spooled printing from disk, any unrecoverable disk error now causes the TurboDOS de-spooler to abort the print job in process and to place the printer in "offline" status. Any remaining print jobs remain queued. No diagnostic message is displayed (since the de-spooler is not attached to any console).

4. **Enhancement to AUTOLOAD command:**
The AUTOLOAD command now accepts multi-command strings separated by a special delimiter, the vertical bar character "|". The AUTOLOAD command converts each vertical bar "|" to a backslant "\" as it is processing the command string. In order to work with release 1.22, **all .AUT files created with AUTOLOAD under prior releases must be regenerated!**

5. **Enhancement to all commands:**
All TurboDOS commands can now accept interactive input from a DO-file.

6. **Enhancement to Flush/Free Buffers (function 115):**
This function now accepts the following parameters:
C = 115
E = drive (0="A", 1="B",..., 15="P")
D = option flags:
bit 7 = 1 to free buffers
bit 6 = 1 to free buffer after disk error "abort" response
bit 5 = 1 to continue after disk error "abort" response
bit 4 = 1 to return after disk error "abort" response

7. **Error in MP/M-format Set Date/Time (function 104) fixed:**
This error caused the system time to be set incorrectly by this function.

8. **Error in RELCVT command fixed:**
This error caused incorrect conversion to occur in some cases where multiple modules were assembled from one source file using the PSA assembler ".PRGEND" pseudo-op.

9. **Error in Compute File Size (function 35) fixed:**
This error caused an incorrect file size to be computed on a file that had been extended and not closed, and resulted in problems with shared files under RM-Cobol.

10. **Error in COPY command fixed:**
This error caused a spurious diagnostic message "Unable to Read Source File" when copying a file which was an exact multiple of 16K in length.

11. **Error in FILSUP module fixed:**
This error caused incorrect allocation on drives with certain numbers of allocation blocks.

12. **Error in NETSVC module fixed:**
This error caused improper operation of direct printing when implicit network forwarding was involved.

13. **Error in NETREQ module fixed:**
This error caused failure to properly diagnose accesses to disk drives marked invalid (0FFH) in the disk assignment table (DSKAST). A "Not Ready Error" is now properly diagnosed.

14. **Error in SPOOLR module fixed:**
This error caused creation of a new print file to interfere with proper operation of functions 17 and 18.

15. **Error in FILMGR module fixed:**
This error caused Rename File (function 23) to fail on files larger than 528K.

16. **Error in NETREQ module fixed:**
This error caused Comm Channel functions (87 through 93) and User-Defined function (127) to fail if the destination processor (specified by DEFDID) had a NETREQ module present.

17. **Error in DSPOOL module fixed:**
This error caused a crash if an attempt was made to change a printer's queue assignment while the printer was in "stopped" mode.

18. **Error in BUFMGR module fixed:**
This error caused many non-reproduceable file and directory errors during disk-intensive multi-user operations. Two known results of this problem are: (1) the same allocation block being assigned to two files simultaneiously, and (2) spurious error conditions being returned during file read, write or close operations. (A patch was issued for release 1.21 to correct this error.)

19. **Error in DO command fixed:**
This error caused a bad temporary file name to be generated if the DO-file name had a one-character type field.

20. **Enhancement to OSLOAD module:**
A new patchable symbol (SCANDN) has been added to the OSLOAD module. If SCANDN is patched to 0FF hex, then the TurboDOS loader will scan from drive P down to drive A (instead of the normal scan from A to P).

21. **Error in FILSUP module fixed:**
This error caused various problems in connection with certain pathologically-ordered hashed-format directories.

22. **Enhancement to DO command:**
If the DO command is invoked by a privileged user under user-number zero, and if the original DO-file has the global attribute, then the temporary DO$-file is now given the global attribute as well.

23. **Error in FILSUP module fixed:**
This error caused function 17/18 sequences to repeat endlessly on hashed-format directories when a function 17 with an unambiguous file specification was followed by a function 18 with an ambiguous file specification.

24. **Enhancement to Rename File (function 23):**
The Rename File function now resets the archived attribute.

25. **Enhancement to FIFOs:**
Two changes have been made to improve shared access to FIFOs. An exclusive write-lock is no longer gained when writing to a FIFO opened in permissive mode. Access to global FIFOs from non-zero user numbers is now always read/write regardless of the setting of the "global write" compatibility flag.

26. **Addition of several command patch points:**
The following patchable locations now exist in TurboDOS command processors:

AUTOLOAD.COM:

103H	"|"	Special command separator character
104H	"\"	Substitute command separator character

BATCH.COM:

103H	"|"	Special command separator character
104H	"\"	Substitute command separator character
105H	0	Disk for BATCH.DO FIFO file

DIR.COM:

103H	3	Left margin for printed directory (;L option)
104H	"^L"	Clear-screen character for multi-screen directory

DO.COM:

103H	"["	Left parameter delimiter
104H	"]"	Right parameter delimiter

LOGON.COM:

103H	"^L"	Clear-screen character

MASTER.COM:

103H	"^A"	Special attention character
104H	"^S"	Substitute attention character

SEND.COM:

103H	"|"	Special command separator character
104H	"\"	Substitute command separator character