Z8O Programmer's Guide

Copyright Notice

Trademark Notice

TurboDOS is a trademark of Software 2000, Inc., and has been registered in the United States and in most major countries of the free world. CP/M, CP/M Plus, and MP/M are trademarks of Digital Research.

Disclaimer

Software 2000, Inc., makes no representations or warranties with respect to the contents of this publication, and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Software 2000, Inc., shall under no circumstances be liable for consequential damages or related expenses, even if it has been notified of the possibility of such damages.

Software 2000, Inc., reserves the right to revise this publication from time to time without obligation to notify any person of such revision.

ABOUT THIS GUIDE

Purpose

We've designed this Z80 Programmer's Guide to provide the information you need to know in order to write application software to run on Z80-based microcomputers under the TurboDOS operating system. This document explains the theory of operation of each internal facility of TurboDOS. It also describes in detail each TurboDOS function that may be called by an application program.

Assumptions

In writing this guide, we've assumed that you are an experienced assembly-language programmer writing application programs for the Z80 TurboDOS environment. We've also assumed you have read the TurboDOS 1.3 User's Guide, and are therefore familiar with the commands and external features of TurboDOS.

Organization

This guide starts with a section that describes the fundamentals of the TurboDOS environment, with emphasis on the organization of memory and the interface and flow of control between application programs and the operating system.

The next two sections explain TurboDOS internals in more detail. One describes the file system, and the other describes serial I/O.

There are two reference sections that explain each TurboDOS function call in detail. One section describes CP/M-compatible functions supported by TurboDOS, while the other describes functions unique to TurboDOS.

Appendices describe a debugging tool called MONITOR, and summarize the function calls in condensed tabular form. Finally, there is an alphabetical index.

T-FUNCTIONS (Continued)

FUNDAMENTALS

This section introduces you to the TurboDOS environment. Emphasis is given to the organization of memory, and to the interface and flow of control between application programs and the operating system. Subsequent sections describe the file system and other facilities in detail.

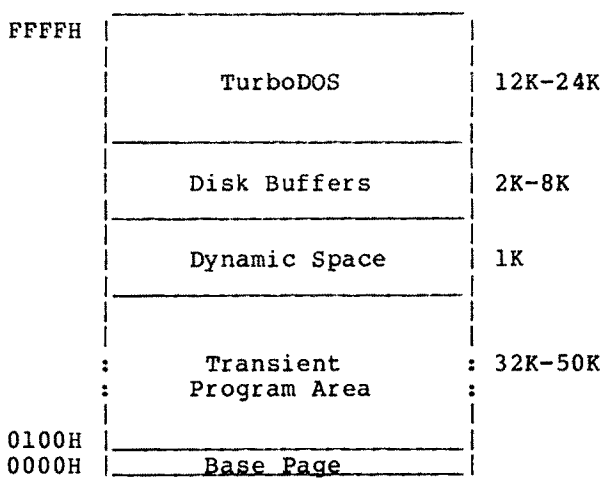
Memory Organization

The resident portion of TurboDOS typically occupies between 12K and 24K of memory, depending on configuration. An additional 2K to 8K (or more) may be devoted to disk buffers, and another 1K or so to print queues, file interlocks, and various other items of dynamic working storage. A 256-byte "Base Page" is reserved at the bottom of memory for communications between TurboDOS and application programs. The remaining memory space available for use by commands and application programs is known as the "Transient Program Area" (TPA).

Z80 TurboDOS supports two alternative memory organization schemes, depending on whether or not bank-switched memory is available.

Non-Banked Memory

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

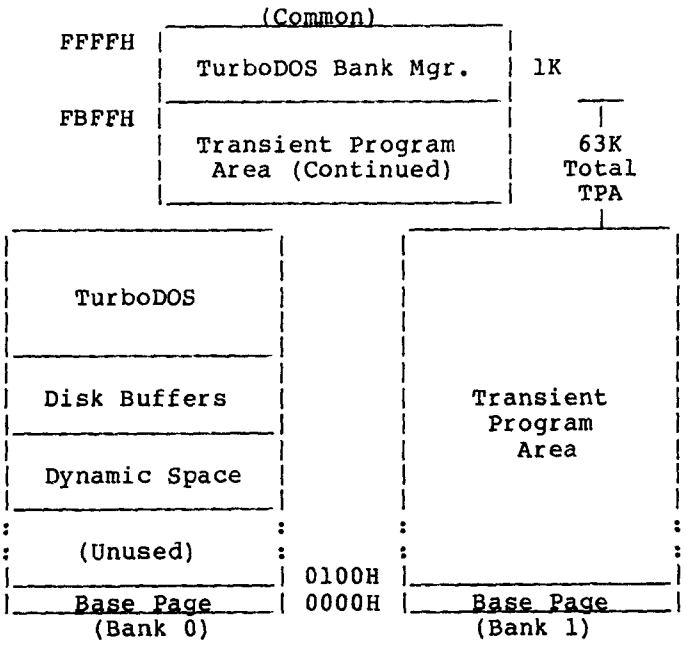


In non-banked systems, the TPA is limited by the size of TurboDOS, its disk buffers, and other dynamic space. If more TPA is needed, the size of the disk buffer area may be decreased using the BUFFERS command or a system call. However, this works only when the system is quiescent (no files open, no print jobs queued, etc.) so that there is no dynamic space allocated below the disk buffers.

Banked Memory

A banked Z80 configuration permits more than 64K of memory by providing two memory banks, only one of which may be active at a time. TurboDOS resides in bank 0 along with its disk buffers and other dynamic space, while a 63K Transient Program Area occupies bank 1.

An area of common (non-switched) memory must be provided at the top of memory. Ideally, it should be 1K, permitting each memory bank to be 63K. However, hardware design often dictates that the common area be larger, often as big as 16K. A large common area reduces the memory available for TurboDOS, but does not reduce the TPA size:



Program Interface

TurboDOS supports 94 different functions that may be invoked by an application program. Functions are provided for file management, console input/output, printing and spooling, and various other TurboDOS facilities. The last half of this guide is largely devoted to describing each of these functions in detail.

Functions supported by TurboDOS fall into two categories: CP/M-compatible functions, and TurboDOS-unique functions. We will refer to them as "C-functions" and "T-functions", respectively. TurboDOS supports 50 C-functions and 44 T-Functions.

C-Functions

To invoke a C-function, a program executes a CALL to location 0005H in the Base Page with a function number in the C-register. TurboDOS supports all BDOS functions of CP/M 2.2:

0 System Reset	20 Read Sequential
1 Console Input	21 Write Sequential
2 Console Output	22 Make File
3*Raw Console Input	23 Rename File
4*Raw Console Output	24 Return Login Vector
5 List Output	25 Return Current Disk
6 Direct Console I/O	26 Set DMA Address
7 Get I/O Byte	27*Get ALV Address
8 Set I/O Byte	28 Write Protect Disk
9 Print String	29 Get R/O Vector
10 Read Cons. Buffer	30 Set File Attributes
11 Get Console Status	31 Get DPB Address
12 Return Version	32 Get/Set User Number
13 Reset Disk System	33 Read Random
14 Select Disk	34 Write Random
15 Open File	35 Compute File Size
16 Close File	36 Set Random Record
17 Search for First	37 Reset Drive
18 Search for Next	(38-39 reserved)
19 Delete File	40*Write Random 0-Fill

C-Functions (Continued)

These TurboDOS C-functions are compatible with the corresponding functions in CP/M 2.2 except for the four functions marked with an asterisk above. In TurboDOS, C-functions 3 (Raw Console Input) and 4 (Raw Console Output) are compatible with MP/M II rather than CP/M. C-function 40 (Write Random with Zero Fill) is synonymous with 34 (Write Random). C-function 27 (Get ALV Address) performs no operation in TurboDOS, but this function affects only the STAT utility of CP/M which is not normally used with TurboDOS.

In addition to BDOS functions 0-40 supported by CP/M 2.2, a number of additional functions have been implemented in CP/M 3 and MP/M II. TurboDOS provides compatible C-functions for certain of these functions:

42	Lock Record	107	Return Serial No.
43	Unlock Record	108	Get/Set Rtn Code
46	Get Free Space	110	Get/Set Delimiter
47	Chain to Program	111	Print Block
104	Set Date/Time	112	List Block
105	Get Date/Time	152	Parse Filename

However, the following rarely-used CP/M 3 and MP/M II functions are <u>not</u> implemented, and perform no function in TurboDOS:

41	Test and Write	98	Free Blocks
44	Set Multi-Sector	99	Truncate File
45	Set Error Mode	100	Set Dir Label
48	Flush Buffers	101	Get Dir Label
49	Get/Set SCB	102	Read PW Mode
50	Direct BIOS Call	103	Write File XFCB
59	Load Overlay	106	Set Default PW
60	Call RSX	109	Get/Set Cons Mode

T-Functions

To invoke a T-function, a program executes a CALL to location 0050H in the Base Page with a function number in the C-register. A different entrypoint address is used to avoid conflict with C-function number assignments. TurboDOS supports the following T-functions:

0	Reset O/S	22	Phys Disk Access
1	Create Process	23	Set Buffer Parms
2	Delay Process	24	Get Buffer Parms
3	Allocate Memory	25	Lock/Unlock Drive
4	Deallocate Memory	26	Flush/Free Buffers
5	Send Message	27	Get/Set Print Mode
6	Receive Message	28	Sig End-of-Print
7	Set Error Address	29	Get/Set Despl Mode
8	Set Abort Address	30	Queue a Print File
9	Set Date/Time	31	Flush List Buffer
10	Get Date/Time	32	Network List Out
11	Rebuild Disk Map	33	Remote Console I/O
12	Get TurboDOS S/N	34	Get Comm Status
13	Set Compat. Flags	35	Comm Input
14	Log-On/Log-Off	36	Comm Output
15	Load File	37	Set Comm Baud Rate
16	Activate Do-File	38	Get Comm Baud Rate
17	Autoload On/Off	39	Set Modem Controls
18	Send Command Line	40	Get Modem Status
19	Get Alloc Info	41	User-Defined Func.
20	Get Phys Disk Info	42	Reorg Disk Directory
21	Get/Set Drv Status	43	Select Mem. Bank

Termination

A program may terminate by jumping to location 0000H in the Base Page, or by invoking C-function 0 (System Reset). Both methods are entirely equivalent, and cause TurboDOS to terminate the program in TPA and prompt for the next command. A program may also terminate by invoking C-function 47 (Chain to Program), which allows the program to specify the next command to be executed after the program terminates.

BIOS Branch Table

For compatibility with CP/M, TurboDOS provides a full simulated "BIOS Branch Table" to support applications that make direct BIOS calls. The branch table is compatible with CP/M 2.2, and it always begins on a page boundary (multiple of 0100H) as in CP/M.

Offset	Function
base+0	cold start
base+3	warm start
base+6	console status to A-reg
base+9	console input to A-reg
base+12	console output from C-reg
base+15	printer output from C-reg
base+18	raw console output from C-reg
base+21	raw console input to A-reg
base+24	home drive to track zero
base+27	select drive from C-reg
base+30	set track from BC-reg
base+33	set sector from BC-reg
base+36	set DMA address from BC-reg
base+39	read disk sector
base+42	write disk sector
base+45	list status to A-reg
base+48	sector translate from BC to HL

Base Page location 0000H contains a JMP instruction to the second (warm start) entry-point of the branch table. Thus, a program can always determine the base address of the branch table by obtaining the address found at Base Page location 0001H and subtracting 3 from it.

Command Processing

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

TurboDOS comes with more than 30 standard command programs (described in detail in the User's Guide). You can expand the vocabulary of commands simply by storing additional programs on disk. Programs are usually kept in .COM files.

Command Prompt

TurboDOS displays a command prompt on the console whenever it is ready to accept a command. The command prompt is composed of the current user number, the current drive letter, and the } prompt symbol.

Command Format

Each TurboDOS command consists of the file name of the program to be executed, possibly followed by an optional command tail of up to 127 characters. A command may be entered in upper- or lower-case letters, but is converted to upper-case by TurboDOS.

The program name may have an explicit file type, but usually doesn't (TurboDOS assumes .COM). It may also have an explicit drive specification (like "B:") if the program is not on the current drive. You will get an error message if the program file cannot be found on disk, or if the available TPA is not big enough to hold the program.

A special kind of command is used to change the current drive. It consists of a drive specification (like "B:") with no program name.

Tail Parsing

The format of a command tail is determined by the particular program involved. TurboDOS passes the command tail to the program by saving the length of the tail (in characters) at location 0080H of the Base Page, and saving the text of the tail (up to 126 characters) starting at location 0081H. TurboDOS also stores a null (zero byte) immediately following the last character of the command tail. The tail includes all characters following the program name, including leading spaces. If no tail is given in the command, the length stored at 0080H is zero.

If the command tail consists of one or two filenames of the form:

{d:}filename{.typ}

then TurboDOS parses each into File Control Block (FCB) format. The first parsed FCB is saved at location 005CH of the Base Page, and the second parsed FCB is saved at location 006CH. Parsing is done following the procedure described for C-function 152 (Parse Filename).

Command Strings

TurboDOS also accepts strings of commands separated by the character \ (backslant). TurboDOS executes each command in sequence, and re-displays each but the first as it is executed. A command string may not exceed the size of the command buffer, which is normally big enough to accomodate two lines of text.

Batch Processing

TurboDOS supports a batch processing mode in which execution is controlled by a predefined sequence of commands stored in a "do-file" on disk. A do-file is a text file (usually type .DO), each line of which contains a valid TurboDOS command or command string. A do-file may be activated with a DO command, or by invoking T-function 16 (Activate Do-File). A do-file may contain any number of embedded DO commands, and nesting is supported to any reasonable depth.

Automatic Loading

TurboDOS provides a facility for loading any program or executing any command sequence automatically at initial start-up (cold start) or whenever a program terminates (warm start). Autoload at cold-start takes place only if a file named COLDSTRT.AUT is present on the start-up disk. Autoload at warm-start takes place only if a file named WARMSTRT.AUT is present on the current disk. The AUTOLOAD command is the usual way to create these .AUT files.

Alternatively, a program (.COM file) may be autoloaded by renaming it as COLDSTRT.AUT or WARMSTRT.AUT. In this case, however, the autoloaded program must not rely on the contents of the Base Page FCB (at 005CH) and buffer (at 0080H), because they will be left uninitialized by after the autoload.

Base Page Layout

The Base Page is the 256-byte memory region from 0000H to 00FFH, and is used primarily for communication between TurboDOS and programs running in the TPA. The organization of the Base Page is shown below:

Locations	Description
0000H-0002H	Contains a JMP instruction to the warm-start entry-point in the BIOS branch table. A program usually terminates by executing a JMP 0000H. The address at location 0001H can be used to locate the BIOS branch table.
0003H	I/O byte, may be used to control device assignments in some implementations.
0004H	Least-significant 4 bits contains current drive. Most-significant 4 bits contains current user number (modulo 15).
0005H-0007H	Contains a JMP instruction to the TurboDOS C-function entrypoint. A program may invoke a C-function by executing a CALL 0005H with a C-function number in the C-register. The address at location 0006H is highest useable address in the TPA plus one, and may be used by a program to find out how much memory it can use.

Base Page Layout (Continued)

Locations	Description
0008H-003AH	Reserved for interrupt vectors (RST 1 through 7).
003BH-004FH	(Unused, reserved)
0050H-0052H	Contains a JMP instruction to the TurboDOS T-function entrypoint. A program may invoke a T-function by executing a CALL 0050H with a T-function number in the C-register.
0053H-005BH	(Unused, reserved)
005CH-006BH	Default FCB part 1. The first filename argument in a command tail is parsed into this 16-byte area.
006CH-007BH	Default FCB part 2. The second filename argument in a command tail is parsed into this 16-byte area, and must be moved to another location before making use of the default FCB.
007CH	Default FCB current record.
007DH-007FH	Default FCB random record.
0080H-00FFH	Default 128-byte buffer. This area receives the command tail length in 0080H, and the command tail text (up to 126 characters plus a null terminator) in locations 0081H-00FFH.

System Start-Up

To get TurboDOS started, it is necessary to read a copy of the operating system from disk into memory, a process known as "cold start". The exact cold-start procedure depends on the particular hardware involved.

Most TurboDOS implementations use this three-step cold-start procedure:

1. When the computer is turned on or reset, it executes the TurboDOS bootstrap from read-only memory (ROM). (In the HORIZON implementation, the bootstrap is loaded from reserved tracks on disk.) The bootstrap scans all disk drives from A to P, searching the directory of each ready drive for a file named OSLOAD.COM which contains the TurboDOS loader. When this file is found, the bootstrap loads it into the TPA and executes it.

2. The TurboDOS loader scans all disk drives from A to P, searching for a file named OSSERVER.SYS which contains the server operating system. When this file is found, the loader proceeds to load the operating system into the upper portion of memory, then transfers control to it. The drive from which the OSSERVER.SYS file was loaded becomes the "system disk".

3. The server downloads a user bootstrap routine into each user processor. The server then locates a file named OSUSER.SYS on the system disk which contains the user operating system, and downloads it into each user processor.

During network operation, it is helpful if the system disk is always on-line. If a fixed disk is available, it should be used as the system disk.

Summary

This section has introduced the fundamentals of the TurboDOS environment. You have learned how memory is organized in both non-banked and banked systems. You understand the TurboDOS program interface, including C-functions, T-functions, and direct BIOS calls. You know how TurboDOS parses and processes commands, command strings, and do-files, and how it communicates with programs via the Base Page.

Next, we examine the TurboDOS file system in considerable detail.

FILE SYSTEM

This section describes the TurboDOS file system in detail. It covers the structure of disks and files, the facilities provided to manage files, and the procedures for calling these facilities from application programs.

Disk Capacity

The TurboDOS file system can support up to sixteen logical drives per processor, identified by the letters A through P. Drives may be local to the processor, or may be attached to another processor and accessed by means of networking.

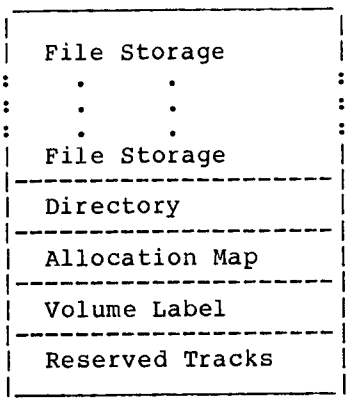
TurboDOS accomodates any combination of drives from mini-floppies to large hard disks in excess of a gigabyte. Allocation block size may be chosen individually for each drive, and affects maximum drive capacity as follows:

Alloc. Block Size	Max. Drive Capacity
1K	256 Kilobytes
2K	128 Megabytes
4K	256 Megabytes
8K	512 Megabytes
16K	1,024 Megabytes

Because these limits are so big, it is almost never necessary to partition a physical drive into smaller logical drives under TurboDOS. However, such partitioning is sometimes done for user convenience when using large fixed disks.

North Star TurboDOS recognizes four standard mini-floppy disk formats: double and quad density (35 track and 70 track) CP/M format, and double and quad density (non-interleaved) TurboDOS format. In addition, 80-track TurboDOS diskettes, although non-standard, can be handled on a read-only basis by the system.

Disk Organization Each disk is organized into five areas:



Reserved tracks are required by certain hardware configurations to support cold-start, but are not otherwise used by TurboDOS. The volume label permits a name to be given to each disk. The allocation map contains one bit for each allocation block on the disk, and is used by TurboDOS to keep track of which disk blocks are occupied and which are free. The directory is a table of contents which identifies all files stored on the disk. The remainder of the disk (most of it) is available for file storage.

CP/M does not maintain a volume label or allocation map on the disks it creates. When a CP/M disk is first accessed by TurboDOS, the first few CP/M directory entries are automatically relocated to the end of the directory in order to make room for the label and map. When a TurboDOS disk is accessed by CP/M, the label and map appear to be ordinary deleted directory entries. Thus, disks can be moved freely between CP/M and TurboDOS in spite of the differences in organization.

Directory Formats

TurboDOS supports two alternative directory formats: linear and hashed. A flag bit in the directory label indicates which format is in use on a particular disk.

The standard linear format is compatible with CP/M, and is searched sequentially. Consequently, look-up speed deteriorates with increasing directory size, and can get painfully slow on large disks with many files.

The optional hashed directory format uses a hashing algorithm to make look-up in large directories much faster. A hashed directory may be used on any disk, but is especially suited for use on hard disks with many files. Hashed directories are not media-compatible with CP/M, but may be converted to linear format whenever exporting to CP/M is needed.

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

File Organization

A file contains a sequence of 128-byte records, and may be up to 134 megabytes (1,048,576 records) long. The records of a file may be read and written sequentially or randomly (by relative record number). A file may be extended by writing beyond the end of file. TurboDOS automatically allocates disk space when a file is extended, and deallocates it when a file is deleted.

Text files are written as a sequence of ASCII characters with a carriage-return (0DH) and line-feed (0AH) at the end of each text line. Text lines are variable length and may span records. The end of a text file is marked by the ASCII character SUB (1AH).

File Operations

About half of the 50 C-functions supported by TurboDOS are connected with the file system. These functions support the operations needed to manipulate files, directories, and disks.

The following functions provide the basic facilities for sequential file access:

C-Fcn	Function Name
15	Open File
16	Close File
20	Read Sequential
21	Write Sequential
22	Make File

These additional functions are necessary to support random access and file sharing:

C-Fcn	Function Name
33	Read Random
34	Write Random
35	Compute File Size
36	Set Random Record
42	Lock Record
43	Unlock Record

Directory functions include:

C-Fcn	Function Name
17	Search for First
18	Search for Next
19	Delete File
23	Rename File
30	Set File Attributes

File Operations (Continued)

Drive-oriented functions are:

C-Fcn	Function Name
14	Select Disk
24	Return Login Vector
25	Return Current Disk
28	Write Protect Disk
29	Get R/O Vector
31	Get DPB Address
37	Reset Drive
46	Get Free Space

Finally, some other functions connected with the file system include:

C-Fcn	Function Name
13	Reset Disk System
26	Set DMA Address
32	Set/Get User Number
47	Chain to Program
152	Parse Filename

Each of these file system C-functions is described in detail later in this document.

Naming Files

TurboDOS keeps track of files by name, maintaining a directory of files on each disk. A file is identified uniquely by four fields:

- drive letter (A-P)
- user number (0-31)
- file name (up to 8 characters)
- file type (up to 3 characters)

The drive letter specifies the disk on which the file is located. If no drive letter is given, the current drive is assumed by default.

The user number specifies one of 32 logical file libraries on each disk. These libraries allow files to be conveniently segregated by user or application. Generally, user 0 is reserved for global files and user 31 is reserved for log-on security, leaving 1-30 for general use.

The name and type fields are composed of ASCII characters. The file name may have up to eight characters, and the file type may have up to three. Shorter names and types are padded on the right with spaces.

It is suggested that file names and file types be composed from the upper-case letters A-Z and the digits 0-9. Actually, any ASCII characters may be used including lower-case letters, punctuation, and even non-printing control characters. However, such names may not be parsed correctly in commands nor displayed correctly in directories.

The question mark ? is a special wild-card character which may be used in file names and types to match any character in the corresponding position during directory searches.

Special File Names

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

File Control Block

File-oriented C-functions and T-functions are always called with the address of a File Control Block (FCB) in the DE-register. The FCB is a data structure 33 bytes long (36 bytes for random access operations) organized as follows:

Offset	Field	Description
0	drive	drive code (0-16): 0 -> current drive 1 -> drive A 2 -> drive B : 16 -> drive P
1-8	name	file name in ASCII, padded on right with spaces, high-order bit of each byte reserved for attributes f1-f8
9-11	type	file type in ASCII, padded on right with spaces, high-order bit of each byte reserved for attributes t1-t3
12	extent	least significant five bits of extent number

File Control Block (Continued)

Offset	Field	Description
13	spec1	flag byte (Do Not Use)
14	spec2	most significant eight bits of extent number
15	record count	number of records in current extent (0-128)
16-31	map	allocation map of current extent
32	current record	current record number (0-127) in current extent
33-35	random record	20-bit record number (byte 33 is least significant) for random-access operations

In general, the application program must initialize FCB bytes 0-12 before opening, making, or searching for a file. It must also zero FCB byte 32 before reading or writing a file sequentially from the beginning.

When a file is opened, TurboDOS fills FCB bytes 0-31 with information from the directory. Thereafter, the application program should not modify FCB bytes 0-31. When the file is closed, TurboDOS updates the directory with information from the FCB. A directory entry has the same structure as the first 32 bytes of an FCB. In a directory entry, however, byte 0 contains the user number 0-31 to which the file belongs, or the value 0E5H if the directory entry is not in use. Also, byte 13 may contain the exact byte count of the last record in the file.

File Attributes

File attributes are stored in the high-order bits of the FCB name field bytes f1-f8 and type field bytes t1-t3, and are used to control how a file may be accessed:

Attribute	Definition
f1	FIFO file attribute
f2-f4	undefined file attributes
f5-f8	interface attributes
t1	read-only file attribute
t2	global file attribute
t3	archived attribute

The file attribute bits f1-f4 and t1-t3 are recorded in the directory, and may be set or cleared by means of C-function 30 (Set File Attributes). For a newly-created file, all attribute bits are initialized to zero. When a file is opened, its attributes are copied into the FCB. File attributes may also be interrogated by means of C-functions 17 and 18 (Search for First/Next).

The read-only attribute (t1) prevents a file from being written, deleted or renamed. The global attribute (t2) enables a file saved under user 0 to be accessed from any user number (it has no effect for files saved under non-zero user numbers). The archived attribute (t3) is used for incremental file backup, and is automatically cleared by TurboDOS whenever a file is written or renamed. The FIFO attribute (f1) causes a file to be accessed using a special "first-in first-out" access method (described later).

Attributes f2-f4 are undefined, and available to the user. Interface attribute bits f5-f8 cannot be used as file attributes; they specify options for certain C-functions.

User Numbers

TurboDOS provides 32 file libraries on each disk corresponding to user numbers 0-31. Generally, user 0 is reserved for global files and user 31 is reserved for log-on security, leaving 1-30 for general use.

The current user number is established initially at log-on. For a non-privileged log-on, the user number remains unchanged until log-off. This restricts file access to the corresponding file library (plus global files under user 0). For a privileged log-on, the user number may be changed without restriction by means of C-function 32 (Set/Get User Number).

The current user number is treated as a prefix to file names, thereby allowing each disk directory to contain up to 32 libraries. Most directory functions (make, rename, delete, search, etc.) are restricted to the library corresponding to the current user number. However, files in the user 0 library which have the global file attribute may be opened from any user number. This permits commands, programs, and other common files to be shared by all users.

File Sharing

In a multi-user TurboDOS system, it is possible for multiple users to access the same file at the same time. This can happen if the users are logged-on to the same user number, or accessing the same global file. TurboDOS supports interlocks to regulate such file sharing at the file or record level.

TurboDOS file sharing facilities are compatible with MP/M, but provide significant extensions to alleviate the most serious deficiencies in MP/M file sharing.

File Locks

File-level interlocks are supported by means of four distinct modes of opening a file. The open mode is determined by FCB interface attributes f5-f6 when the file is opened or created. The four open modes are called exclusive, shared, read-only, and permissive.

A file opened in exclusive mode is available to the opening process exclusively until it is closed, and may not be opened by any other process. A file cannot be opened in exclusive mode if the file is currently opened (in any mode) by another process.

A file may be opened in shared mode by any number of processes simultaneously. All processes are allowed to read, write and extend the file. Record lock and unlock functions are honored only for file opened in shared mode.

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

A file may be opened in permissive mode by any number of processes simultaneously. 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, preventing any other process from writing to the file. The exclusive write-lock is released when the locking process closes the file.

In shared and permissive modes, if a process extends a file by adding new records at the end, these records become immediately accessible to other processes that also have the file open.

Record Locks

Record-level interlocks are controlled by means of explicit locking and unlocking requests made by the application program. This allows concurrent update by multiple processes.

Record locks are by no means automatic, and require explicit cooperative participation by all updating programs. C-functions 42 (Lock Record) and 43 (Unlock Record) are honored only for files opened in the shared mode. Each program 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 code and the program must try again until it is successful. Alternatively, the program can ask TurboDOS to suspend program execution automatically until the lock request can be satisfied.

To extend a shared file in a concurrent update environment, the extending program should first acquire a lock on record N+1 (where N is the last record in the file). The program may then safely write record N+1, and finally unlock N+1.

Compatibility Modes

The file sharing facilities of TurboDOS are designed to provide compatibility with MP/M, 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 some of these rules. To this end, TurboDOS provides a byte of "compatibility flags" with the following bit assignments:

Bit	Flag Name	Affects
7	permissive	default open mode
6	suspend	lock conflict action
5	global-write	writing global files
4	mixed-mode	mixed file open modes
3	logical	record lock validity
2-0	(not defined)	

For each compatibility flag, a zero-bit denotes strict adherence to the 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 its compatibility flags by calling T-function 13 (Set Compatibility Flags), but the flags automatically revert to their initial setting when the program terminates.

If the permissive flag (bit 7) is set, the default file open mode is permissive, rather than exclusive (as in MP/M). Specifically, the open mode is determined when a file is opened or created by FCB interface attributes f5-f6, as shown in the following table:

Compatibility Modes (Continued)

permissive flag = 0			permissive flag = 1		
f6	f5	open mode	f6	f5	open mode
0	0	exclusive	0	0	permissive
0	1	shared	0	1	shared
1	0	read-only	1	0	read-only
1	1	permissive	1	1	exclusive

If the suspend flag (bit 6) is set, then an attempt to lock a record that is already locked by someone else causes the process to be suspended until its 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 return code (as in MP/M).

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

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

If the logical flag (bit 3) is set, then the FCB random record field for C-functions 42 and 43 (Lock/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 field for C-functions 42 and 43 is interpreted as the relative number of a 128-byte record, and causes the file to be positioned to that record (as in MP/M).

FIFO Files

To facilitate communications between processes, processors and users, TurboDOS supports a special kind of file called a FIFO (first-in, first-out) similar in concept to a Unix pipe. 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 (f1) in the directory. Record zero of a FIFO is a header record used by TurboDOS to keep track of the FIFO, and is organized as follows:

Offset	Contents
0	type (0=RAM, -1=disk)
1	mode (0=error code, -1=suspend)
2-3	maximum size (records)
4-5	current size (records)
6-7	number of last record read
8-9	number of last record written
10-127	(not used, reserved)

The header specifies whether the body of the FIFO is RAM- or disk-resident, and the maximum number of records it may contain. RAM-resident FIFOs provide high-speed but limited capacity (up to 127 records, usually much less). 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.

FIFO Files (Continued)

Normally, reading from an empty FIFO returns an end-of-file code (A=1), and writing to a full FIFO returns a disk-full code (A=2). However, if the mode byte in the FIFO header is set to -1 (suspend), 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 using C-functions 33 and 34 (Read/Write Random), thereby bypassing the normal first-in first-out protocol. An attempt to make (C-function 22) an existing FIFO is treated as an open (C-function 15), while an attempt to delete (C-function 19) a FIFO is ignored. The only way to get rid of a FIFO is first to clear the FIFO attribute, then delete it.

Load Optimization

TurboDOS contains a program load optimizer which greatly improves the speed of program loading and overlay fetching from local disk drives. This module scans the allocation map of program files to be loaded into TPA, detects sequentially-allocated file segments (often 16K or more), 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 128-byte record 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 T-function 15 (Load File). The optimizer is not invoked when loading from a remote drive over the network, nor when loading into a banked TPA.

Buffer Management

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

The number and/or size of disk buffers may be changed by means of T-function 23 (Set Buffer Parameters), and interrogated by T-function 24 (Get Buffer Parameters). The number of buffers must be at least two, and the buffer size must be at least as large as the physical sector size of the disks being used. For optimum performance, the number of buffers should be as large as possible consistent with the TPA size required.

The buffer manager maintains its buffers on two 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 sector is already in a buffer. Most of the time it is, and no physical disk access is required. If not, the buffer manager 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 to receive the newly requested disk sector.

Media Changes

Before a removable disk volume is changed, it is crucial that any buffers relating to that disk are written out if necessary, and returned to the free list. In single-user configurations of TurboDOS, this is done automatically whenever the system pauses for console input. In multi-user configurations, buffers must be explicitly flushed and freed by calling T-function 26 (Flush/Free Buffers) prior to changing disks. This is most commonly done by executing the CHANGE command, but should also be coded into applications that require media changes during operation. For safety, TurboDOS also flushes buffers automatically during any lull in system activity, and frees them automatically whenever a disk drive becomes not-ready.

Error Handling

In the event of an unrecoverable disk error, TurboDOS normally displays a diagnostic message in one of these 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]

Spooler Error [Ignore, Abort]
```

and waits for the user to choose the desired recovery option by keying in the appropriate letter (R, I or A).

Error Handling (Continued)

An application program may elect to intercept and process such errors, however, by calling T-function 7 (Set Error Address). In this case, TurboDOS does not display its usual diagnostic messages. Normal error processing resumes automatically when the application program terminates.

NOTE: Because the buffer manager optimizes disk write operations by deferring them as long as possible, write errors may be reported later than expected and possibly even to a different user than expected.

SERIAL I/O

This section describes the TurboDOS facilities that deal with serial input/output (I/O) in connection with consoles, printers, and communications channels.

Console I/O

TurboDOS provides ten C-functions that permit programs to interact with the user console device. Three kinds of console input/output are supported in TurboDOS: basic I/O, raw I/O, and string I/O.

Basic Console I/O

Three C-functions provide basic console I/O on a single-character basis:

C-Fcn	Function Name
1	Console Input
2	Console Output
11	Get Console Status

The Console Input function waits for a character to be keyed in, echoes the character to the console screen to provide visual confirmation, and returns the character to the calling program.

The Console Output function displays a character on the console screen. It expands horizontal tab characters into spaces, based upon tab stops at every eighth column.

The Get Console Status function checks to see whether or not a console input character is available, and returns a Boolean result.

Raw Console I/O

Three additional C-functions provide raw console I/O:

C-Fcn	Function Name
3	Raw Console Input
4	Raw Console Output
6	Direct Console I/O

The Raw Console Input function is similar to the basic Console Input function, except that input characters are not echoed to the screen. Likewise, the Raw Console Output function is like the basic Console Output function, except that horizontal tabs are not expanded.

The Direct Console I/O function combines the functions of Raw Console Input, Raw Console Output, and Console Status. It is supported only for compatibility with CP/M.

String Console I/O

The remaining console I/O functions provide input and output of character strings:

C-Fcn	Function Name
9	Print String
10	Read Console Buffer
110	Get/Set Delimiter
111	Print Block

The Print String function outputs a string of characters to the console. The string may be of any length, and is terminated by a reserved delimiter. The delimiter is normally the dollar-sign $ character, but may be changed by means of the Get/Set Delimiter function.

String Console I/O (Continued)

The Print Block function is similar to Print String, except that the string length is passed explicitly so that no delimiter is needed. Both Print String and Print Block expand horizontal tabs.

The Read Console Buffer function reads an entire line of edited input from the console. Characters are accepted from the console and stored in successive memory locations until a carriage-return terminates the line. Input characters are echoed to console output (but, unlike CP/M, tabs are not expanded). Rudimentary editing is supported: backspace or delete characters erase the last typed character, while CTRL-U or CTRL-X erase the entire line.

Attention Requests

The execution of a program or do-file may be suspended at any time by typing a reserved "attention" character on the console keyboard. In North Star TurboDOS, this is BREAK. TurboDOS will "beep" to acknowledge that it has received the attention request.

After an attention request, the interrupted program or do-file will remain suspended until one of the following attention responses is typed:

CTRL-Q (resume) simply restarts execution at the point of interruption.

CTRL-C (abort) cancels execution of the interrupted program or do-file, causes any nested commands and do-files to be disregarded, and returns to the command prompt. An application program may elect to intercept such abort requests, however, by calling T-function 8 (Set Abort Address).

Attention Requests (Continued)

CTRL-P (echo-print) restarts execution and causes all subsequent console output also to be echoed to the printer. A second attention/echo sequence turns off echoing of console output to the printer.

CTRL-L (end-print) restarts execution after signalling the end of the current print job.

Comm Channel I/O

In order to allow communications-oriented applications programs to be written in a hardware-independent fashion, TurboDOS supports a standard communications channel interface consisting of seven T-functions:

T-Fcn	Function Name
34	Get Comm Channel Status
35	Comm Channel Input
36	Comm Channel Output
37	Set Comm Channel Baud Rate
38	Get Comm Channel Baud Rate
39	Set Comm Channel Modem Controls
40	Get Comm Channel Modem Status

These functions support multiple channels of communications. T-functions 34-36 provide basic single-character comm channel I/O (analogous to raw console I/O). T-functions 37-38 allow programs to sense or set the comm channel baud rate to any standard speed from 50 to 19,200 baud. T-functions 39-40 allow programs to set modem control signals (RTS, DTR) and to sense modem status signals (CTS, DSR, DCD, RI).

Printer Output

TurboDOS provides the basic printing functions of CP/M, plus an elaborate concurrent printing facility which offers several modes of print spooling and flexible print routing among multiple printers and print queues. The spooling and routing facilities are completely transparent to application programs.

Basic Printing

Two C-functions provide the basic means for programs to generate printer output:

C-Fcn	Function Name
5	List Output
112	List Block

The List Output function outputs a single character to be printed, while the List Block function outputs a character string of specified length. In contrast to console I/O, these print output functions do not expand tabs.

Control Functions

Four T-functions provide control over the print spooling, de-spooling, and queuing mechanisms of TurboDOS:

T-Fcn	Function Name
27	Get/Set Print Mode
28	Signal End-of-Print
29	Get/Set De-Spool Mode
30	Queue a Print File

The Get/Set Print Mode function controls print routing. Print output may be routed direct to a specified printer, spooled to a specified drive and print queue, displayed on the console, or simply discarded.

Control Functions (Continued)

The Signal End-of-Print function allows a program to terminate a print job explicitly. In the absence of this function, a print job ends automatically at the conclusion of the program, upon receipt of an end-print attention request from the console, or when a reserved end-of-print character (if defined) appears in the print output stream.

The Get/Set De-Spool Mode function controls background printing (de-spooling). A printer may be assigned to de-spool from a specified queue, or may be placed in an off-line status. Any print job in process may be stopped, resumed, restarted from the beginning, or terminated altogether.

The Queue a Print File function permits a program to queue a print file (or any text file, for that matter) for background printing. The file may be placed on any specified print queue, and may be saved or deleted automatically after printing.

C-FUNCTIONS

This section describes the 50 CP/M-compatible functions ("C-functions") supported by TurboDOS. The C-functions are presented in numerical order, with calling parameters, return value, and a detailed explanation for each.

To invoke a C-function, a program executes a CALL to location 0005H in the Base Page with a function number in register C. Byte-length arguments are passed in register E, and word-length arguments in register DE. C-functions return byte-length values in register A (duplicated in L), or word-length values in register HL (duplicated in BA).

If a C-function call is made with register C set to an unsupported function number, TurboDOS returns immediately with registers HL and BA zeroed.

C-function calls generally destroy registers A-B-C-D-E-H-L, but preserve X, Y, and the alternate register set.

C-Function 0 **System Reset**

Entry Arguments

Reg	Description
C	= 0

Explanation

The System Reset function terminates the calling program ("warm-start"). Program termination is more commonly performed by executing a jump to location 0000H, which has exactly the same effect.

In a multi-user TurboDOS system, program termination closes any open files, releases any locked records or devices, and ends any active print job.

C-Function 1 **Console Input**

Entry Arguments

Reg	Description
C	= 1

Returned Value

Reg	Description
A	= input character

Explanation

The Console Input function obtains the next character from the console keyboard, and returns it in register A. If no character is available, the calling program is suspended until a character is typed.

Graphic characters and certain control characters (carriage-return, line-feed, and backspace) are echoed to the console screen. Horizontal tabs are expanded into multiple spaces, based upon tab stops at every eighth column.

C-Function 2 **Console Output**

Entry Arguments

Reg	Description
C	= 2
E	= output character

Explanation

The Console Output function displays the character passed in register E on the console screen. Horizontal tabs are expanded into multiple spaces, based upon tab stops at every eighth column.

C-Function 3 **Raw Console Input**

Entry Arguments

Reg	Description
C	= 3

Returned Value

Reg	Description
A	= input character

Explanation

The Raw Console Input function obtains the next character from the console keyboard, and returns it in register A. If no character is available, the calling program is suspended until a character is typed. Input characters are not echoed to the console screen.

This function is compatible with MP/M. (In CP/M 2.2, this function is Input from Reader Device. In CP/M 3, this function is Auxiliary Input.)

C-Function 4 **Raw Console Output**

Entry Arguments

Reg	Description
C	= 4
E	= output character

Explanation

The Raw Console Output function displays the character passed in register E on the console screen. Horizontal tabs are not expanded.

This function is compatible with MP/M. (In CP/M 2.2, this function is Output to Punch Device. In CP/M 3, this function is Auxiliary Output.)

C-Function 5 **List Output**

Entry Arguments

Reg	Description
C	= 5
E	= output character

Explanation

The List Output function sends the character passed in register E to be printed according to the current print routing. Horizontal tabs are not expanded.

C-Function 6 **Direct Console I/O**

Entry Arguments

Reg	Description
C	= 6
E	= -1 (for combined status/input) -2 (for status) -3 (for raw input) output character (for raw output)

Returned Value

Reg	Description
A	= input character or status

Explanation

The Direct Console I/O function performs one of four possible sub-functions, depending upon the argument passed in register E.

If E = -1 (0FFH), then any available console input character is returned in register A, without echo to the screen. If no character is available, the function returns A = 0.

If E = -2 (0FEH), then this function returns console status (A = 0 if no console input is available, or A = -1 otherwise). Equivalent to C-function 11 (Get Console Status).

If E = -3 (0FDH), then this function obtains the next console input character and returns it in register A, without echo to the screen. If no character is available, the calling program is suspended until a character is typed. Equivalent to C-function 3 (Raw Console Input).

For other values of E, this function displays the character on the console screen. Horizontal tabs are not expanded. Equivalent to C-function 4 (Raw Console Output).

C-Function 7 **Get I/O Byte**

Entry Arguments

Reg	Description
C	= 7

Returned Value

Reg	Description
A	= contents of I/O byte (loc 0003H)

Explanation

This function simply returns the value of Base Page location 0003H (which is used in some implementations to control serial I/O device assignment).

NOTE: This function is supported only if the optional module CPMSUP is included during TurboDOS system generation.

C-Function 8 **Set I/O Byte**

Entry Arguments

Reg	Description
C	= 7
E	= new value of I/O byte (loc 0003H)

Explanation

This function simply sets the value of Base Page location 0003H (which is used in some implementations to control serial I/O device assignment).

NOTE: This function is supported only if the optional module CPMSUP is included during TurboDOS system generation.

C-Function 9 **Print String**

Entry Arguments

Reg	Description
C	= 9
DE	= string address

Explanation

The Print String function displays a string of characters on the console screen. The string may be of any length, and is terminated by a reserved delimiter. The delimiter is normally the dollar-sign $ character, but may be changed by means of C-function 110 (Get/Set Output Delimiter). Horizontal tabs are expanded into multiple spaces, based upon tab stops at every eighth column.

C-Function 10 **Read Console Buffer**

Entry Arguments

Reg	Description
C	= 10
DE	= buffer address

Explanation

The Read Console Buffer function reads an entire line of edited input from the console. The input buffer whose address is passed in register DE has the following structure:

Offset	Direction	Description
0	passed	max input size (N)
1	returned	actual input (0-N)
2 to N+1	returned	input characters

The first byte of the buffer must be preset to the maximum number of characters allowed in the input line.

Console input is accepted until terminated by a carriage-return. Input errors may be corrected by typing BACKSPACE or DELETE to erase one character at a time, or CTRL-U or CTRL-X to erase the entire line. Characters in excess of the maximum are not accepted, and diagnosed with a "beep". Input characters are echoed to the console screen. Unlike CP/M, this function does not expand tabs in TurboDOS.

Upon return, the second byte of the buffer contains the actual number of input characters in the buffer. The input line is returned starting at the third byte of the buffer. The terminating carriage-return is neither stored in the buffer nor included in the count. Unused buffer positions following the last input character are uninitialized.

C-Function 11 **Get Console Status**

Entry Arguments

Reg	Description
C	= 11

Returned Value

Reg	Description
A	= -1 if console input is available 0 if console input is not available

Explanation

The Get Console Status function checks to see whether or not a console input character is available. If console input is available, it returns A = -1. Otherwise, it returns A = 0.

C-Function 12 **Return Version**

Entry Arguments

Reg	Description
C	= 12

Returned Value

Reg	Description
H	= 00H (meaning: CP/M, not MP/M)
L	= 31H (meaning: BDOS version 3.1)

Explanation

The Return Version function provides information on the latest compatible version of CP/M. (The BDOS version number returned in register L may be changed by patching the symbol CPMVER during system generation.)

C-Function 13 **Reset Disk System**

Entry Arguments

Reg	Description
C	= 13

Explanation

In TurboDOS, the only effect of the Reset Disk System function is to reset the current DMA address to 0080H. (See C-function 26, Set DMA Address.)

C-Function 14 **Select Disk**

Entry Arguments

Reg	Description
C	= 14
E	= selected disk drive: 0 for drive A 1 for drive B : 15 for drive P

Explanation

The Select Disk function causes the disk drive specified in register E to be selected as the current (default) disk drive. The current drive is used in subsequent file operations whenever the FCB drive field is set to zero.

C-Function 15 **Open File**

Entry Arguments

Reg	Description
C	= 15
DE	= FCB address

Returned Value

Reg	Description
A	= 0 if successful -1 if file not found

Explanation

The Open File function opens the file specified by the FCB drive, name, type, and extent fields (bytes 0 through 12). Normally, the extent field (byte 12) should be set to zero. The specified file must exist under the current user number or must be a global file under user 0.

The open mode is determined by compatibility flag bit 7 (permissive) and by the FCB interface attributes f5 and f6, as shown in the following table:

permissive flag = 0			permissive flag = 1		
f6	f5	open mode	f6	f5	open mode
0	0	exclusive	0	0	permissive
0	1	shared	0	1	shared
1	0	read-only	1	0	read-only
1	1	permissive	1	1	exclusive

If the FCB current record field (byte 32) is set to -1, this function returns the byte count of the last record of the file in the current record field. The calling program should zero the current record field before doing sequential reads or writes.

C-Function 16 **Close File**

Entry Arguments

Reg	Description
C	= 16
DE	= FCB address

Returned Value

Reg	Description
A	= 0 if successful -1 if file not found

Explanation

The Close File function closes a file previously opened by an Open File (15) or Make File (22) C-function. The directory is updated if necessary to reflect any new blocks allocated to the file, and any locked records are unlocked.

If FCB interface attribute f5 is set, this function performs a "partial close" operation which updates the directory but leaves the file open.

C-Function 17 **Search for First**

Entry Arguments

Reg	Description
C	= 17
DE	= FCB address

Returned Value

Reg	Description
A	= entry number (0-3) if successful -1 if file not found

Explanation

The Search for First function scans the directory for the first entry which matches the FCB drive, name, type, and extent fields (bytes 0 through 12) and the current user number. An ASCII question mark (3FH) in any FCB byte 1 through 12 is treated as a wild-card which matches any character in the corresponding byte position of the directory entry.

If the search is successful, this function returns a directory record (containing four 32-byte directory entries) at the current DMA address, and with a value in register A (0-3) that indicates which of the four entries was found to match the FCB. If the search is not successful, the function returns -1 (0FFH) in register A.

If the Search for First function succeeds in finding an entry which matches the given FCB, then C-function 18 (Search for Next) may be called repeatedly to locate all remaining matches in the directory.

Explanation (Continued)

A special situation occurs if the FCB drive field (byte 0) is set to a question mark (3FH). In this case, the remainder of the FCB is ignored, the directory of the current drive is searched, and the Search for First function returns the very first directory entry (usually the volume label). The Search for Next function will then return each successive directory entry in sequence, regardless of user number. Even deleted entries are returned in this case.

C-Function 18 **Search for Next**

Entry Arguments

Reg	Description
C	= 18

Returned Value

Reg	Description
A	= entry number (0-3) if successful -1 if file not found

Explanation

The Search for Next function continues the search initiated by C-function 17 (Search for First). If the search is successful, this function returns a directory record (containing four 32-byte directory entries) at the current DMA address, and with a value in register A (0-3) that indicates which of the four entries was found to match the FCB. If the search is not successful, the function returns -1 (0FFH) in register A.

C-Function 19 **Delete File**

Entry Arguments

Reg	Description
C	= 19
DE	= FCB address

Returned Value

Reg	Description
A	= 0 if successful -1 if no file was deleted

Explanation

The Delete File function deletes the file specified by the FCB drive, name, and type fields (bytes 0 through 11) and the current user number. ASCII question marks (3FH) may be used as wild-cards anywhere in the FCB name and type fields, in which case this function deletes all matching files.

A program may delete a file that it has open, in which case a close is performed implicitly before the file is deleted. However, a program is not permitted to delete a file that another process has open, nor a file that has the read-only or FIFO attributes.

C-Function 20 **Read Sequential**

Entry Arguments

Reg	Description
C	= 20
DE	= FCB address

Returned Value

Reg	Description
A	= 0 if successful
	1 if at end-of-file
	2 if reading unwritten data

Explanation

The Read Sequential function reads the next 128-byte record from a file into memory at the current DMA address. The given FCB must have been previously opened by an Open (15) or Make (22) C-function, and the FCB current record field (byte 32) initialized to zero.

This function uses the FCB extent and current record fields to determine the record to be read, then increments the current record field in preparation for the next sequential operation. If the current record field overflows, the next extent is opened and the current record field is reset to zero.

C-Function 21 **Write Sequential**

Entry Arguments

Reg	Description
C	= 21
DE	= FCB address

Returned Value

Reg	Description
A	= 0 if successful
	1 if file too large (>134 Mb)
	2 if disk full or file read-only
	8 if attempt to write locked record
	-1 if no directory space

Explanation

The Write Sequential function writes the next 128-byte record of a file from the current DMA address in memory. The given FCB must have been previously opened by an Open (15) or Make (22) C-function, and the FCB current record field (byte 32) initialized to zero.

This function uses the FCB extent and current record fields to determine the record to be write, then increments the current record field in preparation for the next sequential operation. If the current record field overflows, the next extent is opened (or created if it does not exist) and the current record field is reset to zero.

C-Function 22 Make File

Entry Arguments

Reg	Description
C	= 22
DE	= FCB address

Returned Value

Reg	Description
A	= 0 if successful -1 if directory full, file exists, or FCB invalid

Explanation

The Make File function creates a new (empty) file specified by the FCB drive, name, type, and extent fields (bytes 0 through 12). Normally, the extent field (byte 12) should be set to zero. The directory entry for the new file is placed under the current user number. All file attributes are initialized to zero. A request to make a file that already exists is denied.

The newly-created file is left in an open state. If the FCB interface attribute f5 is set, then the file is left open in shared mode. Otherwise, the file is left open in either exclusive or permissive mode, depending on compatibility flag bit 7 (permissive).

The calling program should zero the FCB current record field (byte 32) before doing sequential reads or writes on the file.

C-Function 23 **Rename File**

Entry Arguments

Reg	Description
C	= 23
DE	= FCB address

Returned Value

Reg	Description
A	= 0 if successful -1 if file not found, file in-use, or file name invalid

Explanation

The Rename File function renames the file specified by the FCB drive, name, and type fields (bytes 0 through 11) and the current user number. The file is given the new name and type specified in bytes 17 through 27 of the FCB. Wild-card characters (ASCII question marks) are not allowed in either the old or new name. All remaining bytes of the FCB are disregarded by this function.

A program may rename a file that it has open, in which case a close is performed implicitly before the file is renamed. However, a program is not permitted to rename a file that another process has open, nor a file that has the read-only attribute.

C-Function 24 **Return Login Vector**

Entry Arguments

Reg	Description
C	= 24

Returned Value

Reg	Description
HL	= login rector

Explanation

The Return Login Vector function tests the ready status of all disk drives. It returns a 16-bit vector in register HL containing a one-bit for each drive that is ready for access, and a zero-bit for each drive that is not ready or not defined. The least significant bit corresponds to drive A, and the most significant bit to drive P.

NOTE: This function is supported only if the optional module CPMSUP is included during TurboDOS system generation.

C-Function 25 **Return Current Disk**

Entry Arguments

Reg	Description
C	= 25

Returned Value

Reg	Description
A	= current disk drive: 0 for drive A 1 for drive B : 15 for drive P

Explanation

The Return Current Disk function returns the identity of the current (default) disk drive in register A.

C-Function 26 **Set DMA Address**

Entry Arguments

Reg	Description
C	= 26
DE	= DMA address

Explanation

The Set DMA Address function causes the memory address specified in register DE to be used as the record buffer address for subsequent file read and write operations. In a banked memory system, the DMA address is always interpreted as an address in the same memory bank as the TPA.

Whenever a program is loaded into the TPA, the DMA address is initialized to 0080H, the address of the default record buffer in the Base Page. C-function 13 (Reset Disk System) also sets the DMA address to 0080H.

C-Function 27 **Get ALV Address**

Entry Arguments

Reg	Description
C	= 27

Returned Value

Reg	Description
HL	= 0

Explanation

This function performs no operation in TurboDOS. (Under CP/M, it returns the address of the memory-resident allocation vector for the current disk.)

C-Function 28 **Write Protect Disk**

Entry Arguments

Reg	Description
C	= 28

Explanation

The Write Protect Disk function marks the current (default) disk drive as read-only, preventing any program from writing to the disk. C-function 37 (Reset Drive) must be used to enable writes to the disk once again.

Unlike CP/M, TurboDOS does not re-enable writing after warm-start, C-function 0 (System Reset), or C-function 13 (Reset Disk System). Consequently, write-protection of a disk drive is not nearly so temporary as it is in CP/M.

NOTE: This function is supported only if the optional module CPMSUP is included during TurboDOS system generation.

C-Function 29 **Get Read-Only Vector**

Entry Arguments

Reg	Description
C = 29	

Returned Value

Reg	Description
HL = read-only vector	

Explanation

The Get Read-Only Vector function returns a 16-bit vector in register HL containing a one-bit for each disk 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.

NOTE: This function is supported only if the optional module CPMSUP is included during TurboDOS system generation.

C-Function 30 **Set File Attributes**

Entry Arguments

Reg	Description
C = 30	
DE = FCB address	

Returned Value

Reg	Description
A = 0 if successful	
-1 if file not found or in-use	

Explanation

The Set File Attributes function searches the directory for the file specified by the FCB drive, name, and type fields (bytes 0 through 11) and the current user number, and updates the file attributes in the directory from those in the FCB. (File attributes are stored in the high-order bit of FCB bytes 1-4 and 9-11.)

In addition, if FCB interface attribute f6 is set, this function updates the last record byte count of the file. The count is obtained from the current record field (byte 32) of the FCB, and stored in the specl field (byte 13) of each directory entry.

A program may set attributes on a file that it has open, in which case a close is performed implicitly before the attributes are set. However, a program is not permitted to set attributes on a file that another process has open.

C-Function 31 **Get DPB Address**

Entry Arguments

Reg	Description
C	= 31

Returned Value

Reg	Description
HL	= DPB address

Explanation

The Get DPB Address function causes TurboDOS to construct a CP/M-style Disk Parameter Block (DPB) for the current drive, and to return its memory address in register HL.

NOTE: This function is supported only if the optional module CPMSUP is included during TurboDOS system generation.

C-Function 32 **Get/Set User Number**

Entry Arguments

Reg	Description
C	= 32
E	= -1 to get user number 0-31 to set user number

Returned Value

Reg	Description
A	= user number 0-31

Explanation

The Get/Set User Number function can be used either to set or to return the current user number. If the value -1 (0FFH) is passed in register E, this function returns the current user number in register A. If some other value is passed in register E and if the caller is a privileged log-on, this function sets the current user number to the specified value (modulo 32). A request to set the current user number from a non-privileged log-on is ignored.

C-Function 33 **Read Random**

Entry Arguments

Reg	Description
C = 33	
DE = FCB address	

Returned Value

Reg	Description
A = 0	if successful
1	if reading unwritten data
3	if error changing extents
4	if reading unwritten extent
6	if random record number invalid

Explanation

The Read Random function reads a 128-byte record from a file into memory at the current DMA address. The particular record to be read is specified by a 20-bit random record number obtained from FCB random record field (bytes 33 through 35). The given FCB must have been previously opened by an Open (15) or Make (22) C-function.

This function sets the FCB extent and current record fields to correspond with the record that was read. Unlike C-function 20 (Read Sequential), however, it does not increment the current record field after reading. Thus, if the Read Random function is followed by a Read Sequential or Write Sequential, the same record is re-accessed.

C-Function 34 **Write Random**

Entry Arguments

Reg	Description
C	= 34
DE	= FCB address

Returned Value

Reg	Description
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 invalid
	8 if writing locked record

Explanation

The Write Random function writes a 128-byte record to a file from the current DMA address in memory. The particular record to be written is specified by a 20-bit random record number obtained from FCB random record field (bytes 33 through 35). The given FCB must have been previously opened by an Open (15) or Make (22) C-function.

This function sets the FCB extent and current record fields to correspond with the record that was written. Unlike C-function 21 (Write Sequential), however, it does not increment the current record field after writing. Thus, if the Write Random function is followed by a Read Sequential or Write Sequential, the same record is re-accessed.

C-Function 35 **Compute File Size**

Entry Arguments

Reg	Description
C	= 35
DE	= FCB address

Returned Value

Reg	Description
A	= 0 if successful
	-1 if file not found

Explanation

The Compute File Size function searches the directory for the file specified by the FCB drive, name, and type fields (bytes 0 through 11). If the file is found, this function sets the FCB random record field (bytes 33 through 35) to a value one greater than the record number of the last record in the file. Thus, a succeeding Write Random function (34) will append an additional record at the end of the file.

In TurboDOS, the Compute File Size function returns the correct result whether the file is open or closed.

C-Function 36 **Set Random Record**

Entry Arguments

Reg	Description
C	= 36
DE	= FCB address

Explanation

The Set Random Record function returns the current file position of an open file in the random record field (bytes 33-35) of the FCB. (The file position is determined from the values of the FCB extent, spec2, and current record fields.) Since the Read Sequential (20) and Write Sequential (21) functions do not update the random record field of the FCB, this function is useful when switching from sequential to random access.

C-Function 37 **Reset Drive**

Entry Arguments

Reg	Description
C	= 37
DE	= reset vector

Explanation

The Reset Drive function write-enables the disk drives specified by the 16-bit reset vector passed in register DE. The reset vector contains a one-bit for each disk drive that is to be write-enabled, 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.

NOTE: This function is supported only if the optional module CPMSUP is included during TurboDOS system generation.

C-Function 40 **Write Random with Zero Fill**

Entry Arguments

Reg	Description
C	= 40
DE	= FCB address

Returned Value

Reg	Description
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 invalid
	8 if writing locked record

Explanation

The Write Random with Zero Fill function is implemented in TurboDOS as a synonym for Write Random (C-function 34).

C-Function 42 **Lock Record**

Entry Arguments

Reg	Description
C	= 42
DE	= FCB address

Returned Value

Reg	Description
A	= 0 if successful
	1 if positioning to unwritten data
	3 if error changing extents
	4 if positioning to missing extent
	6 if random record number invalid
	8 if locked by another process

Explanation

The Lock Record function attempts to obtain a lock on the record specified by a 20-bit random record number obtained from FCB random record field (bytes 33 through 35). The given FCB must have been previously opened in shared mode. If the file is not open in shared mode, this function performs no operation and returns a successful result.

The file is positioned to the specified record, unless compatibility flag bit 3 (logical) is set. If the specified record is already locked by another process, this function either suspends or returns an error (A=8) depending upon the setting of compatibility flag bit 6 (suspend).

If the FCB random record field 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). In this case, no positioning is performed.

C-Function 43 **Unlock Record**

Entry Arguments

Reg	Description
C	= 43
DE	= FCB address

Returned Value

Reg	Description
A	= 0 if successful
	1 if positioning to unwritten data
	3 if error changing extents
	4 if positioning to missing extent
	6 if random record number invalid

Explanation

The Unlock Record function unlocks the record specified by a 20-bit random record number obtained from FCB random record field (bytes 33 through 35). Attempting to unlock a record which was not previously locked does not return an error. The given FCB must have been previously opened in shared mode. If the file is not open in shared mode, this function performs no operation and returns a successful result.

The file is positioned to the specified record, unless compatibility flag bit 3 (logical) is set.

If the FCB random record field is set to the 24-bit value 0FFFFFFH, then this function releases any all-inclusive lock on the file, but does not affect any individual record locks. In this case, no positioning is performed.

C-Function 46 **Get Disk Free Space**

Entry Arguments

Reg	Description
C	= 46
E	= disk drive: 0 for drive A 1 for drive B : 15 for drive P

Returned Value

Reg	Description
A	= 0

Explanation

The Get Disk Free Space function determines the amount of free space on the specified disk drive. It returns a 24-bit binary value (the number of free 128-byte records) as a three byte quantity stored at the current DMA address, least significant byte first.

C-Function 47 **Chain to Program**

Entry Arguments

Reg	Description
C	= 47
E	= 0 to revert to orig. current disk -1 to retain present current disk

Explanation

The Chain to Program function provides a means of chaining from one program to another. The calling program must place a valid TurboDOS command line, terminated by a null byte, in the Base Page record buffer starting at location 0080H. This function terminates the calling program, and then executes the command line.

If E = 0, the current disk reverts to what it was when the calling program was originally loaded into the TPA (the normal warm-start procedure). If E = -1, however, the current disk at the time of call is retained.

C-Function 104 Set Date and Time

Entry Arguments

Reg	Description
C	= 104
DE	= date/time packet address

Explanation

The Set Date and Time function sets the system date and time. The address of a four-byte date/time packet is passed in DE. The date/time packet has the following structure:

Offset	Description
0-1	Date, represented as a 16-bit Julian date with 0000H corresponding to 31 December 1977.
2	Hours, represented as two binary coded decimal (BCD) digits
3	Minutes, represented as two binary coded decimal (BCD) digits

Seconds are set to zero.

C-Function 105 **Get Date and Time**

Entry Arguments

Reg	Description
C	= 105
DE	= date/time packet address

Returned Value

Reg	Description
A	= seconds (two BCD digits)

Explanation

The Get Date and Time function returns the system date and time in a four-byte date/time packet whose address is passed in DE. The date/time packet has the following structure:

Offset	Description
0-1	Date, represented as a 16-bit Julian date with 0000H corresponding to 31 December 1977.
2	Hours, represented as two binary coded decimal (BCD) digits
3	Minutes, represented as two binary coded decimal (BCD) digits

Seconds are returned in register A, represented as two binary coded decimal (BCD) digits.

C-Function 107 **Return Serial Number**

Entry Arguments

Reg	Description
C	= 107
DE	= address of 6-byte S/N field

Explanation

The Return Serial Number function returns the CP/M serial number in the 6-byte field whose address is passed in DE. Under TurboDOS, this function always returns six zero bytes.

NOTE: This function is supported only if the optional module CPMSUP is included during TurboDOS system generation.

C-Function 108 **Get/Set Program Return Code**

Entry Arguments

Reg	Description
C = 108	
DE = 0FFFFH (if get)	
program return code (if set)	

Returned Value

Reg	Description
HL = program return code (if get)	

Explanation

The Get/Set Program Return Code function provides a means for one program to pass a 16-bit value to another program. For example, this function can be used to advantage in connection with C-function 47 (Chain to Program).

If register DE is set to 0FFFFH, then this function interrogates the program return code and returns it in register HL. Otherwise, this function sets the program return code to the value passed in register DE.

C-Function 110 **Get/Set Program Output Delimiter**

Entry Arguments

Reg	Description
C	= 110
DE	= 0FFFFH (if get), or
E	= output delimiter (if set)

Returned Value

Reg	Description
A	= output delimiter (if get)

Explanation

The Get/Set Output Delimiter function can be used to set or interrogate the output delimiter used by C-function 9 (Print String). Whenever a program is loaded into the TPA, the output delimiter is initialized to the dollar sign $ character.

If register DE is set to 0FFFFH, then this function interrogates the current output delimiter and returns it in register A. Otherwise, this function sets the output delimiter to the value passed in register E.

C-Function 111 **Print Block**

Entry Arguments

Reg	Description
C	= 111
DE	= CCB address

Explanation

The Print Block function displays a string of characters on the console screen. The string may be of any length, and is defined by a Character Control Block (CCB) whose address is passed in register DE. The CCB is four bytes long, and has the following structure:

Offset	Description
0-1	starting address of string
2-3	byte-length of string

Horizontal tabs are expanded into multiple spaces, based upon tab stops at every eighth column.

C-Function 112 **List Block**

Entry Arguments

Reg	Description
C	= 112
DE	= CCB address

Explanation

The List Block function sends a string of characters to be printed according to the current print routing. The string may be of any length, and is defined by a Character Control Block (CCB) whose address is passed in register DE. The CCB is four bytes long, and has the following structure:

Offset	Description
0-1	starting address of string
2-3	byte-length of string

Horizontal tabs are not expanded.

C-Function 152 **Parse Filename**

Entry Arguments

Reg	Description
C	= 152
DE	= PFCB address

Returned Value

Reg	Description
HL	= 0 if successful and end of line 0FFFFH if error while parsing delimiter address otherwise

Explanation

The Parse Filename function parses an ASCII file specification of the form:

{d:}filename{.typ}

into FCB format. The FCB drive, name, and type fields (bytes 0 through 11) are initialized according to the parsed file specification. Bytes 12 through 15 of the FCB are zeroed.

This function is called with the address of a Parse Filename Control Block (PFCB) in DE. The PFCB is four bytes long, and has the following structure:

Offset	Description
0-1	address of ASCII input string
2-3	address of destination FCB

This function parses the first file specification it finds in the input string. Leading spaces are ignored. Parsing stops upon encountering a space, comma, semicolon, equal-sign, or any ASCII control character.

T-FUNCTIONS

This section describes the 44 TurboDOS-unique functions ("T-functions") which supplement the C-functions described in the previous section. The T-functions are presented in numerical order, with calling parameters, return values, and a detailed explanation for each.

To invoke a T-function, a program executes a CALL to location 0050H in the Base Page with a function number in register C. Arguments are passed and values returned in registers, as described below for each T-function.

If a T-function call is made with register C set to an unsupported function number, TurboDOS returns immediately with register A set to zero.

T-function calls generally destroy registers A-B-C-D-E-H-L, but preserve X, Y, and the alternate register set.

T-Function 0 **Reset Operating System**

Entry Arguments

Reg	Description
C	= 0

Explanation

The Reset Operating System function unlocks all locked records, closes all open files, unlocks all locked drives, and terminates any network sessions involving the calling process.

TurboDOS automatically performs this function at each program termination (warm-start), so it is almost never necessary for a program to call this function explicitly.

T-Function 1 **Create Process**

Entry Arguments

Reg	Description
C	= 1
DE	= entrypoint address
HL	= workspace address

Returned Value

Reg	Description
A	= 0 if successful -1 if insufficient memory

Explanation

The Create Process function creates a new process which starts execution at the entry-point address passed in register DE. The new process is assigned a TurboDOS work area whose address appears to the new process in index register X, and a 64-word stack area whose address appears in the stack pointer register. If the process requires a re-entrant work area (usually allocated dynamically using T-function 3), its address should be passed in register HL and will appear to the new process in index register Y.

If this function is called with register DE set to zero, it causes the calling process to terminate.

NOTE: This function is not intended for use by transient programs, and must be used with great care.

T-Function 2 **Delay Process**

Entry Arguments

Reg	Description
C	= 2
DE	= tick count

Explanation

The Delay Process function causes the calling process to be suspended for the period of time specified by the tick count passed in register DE. A system "tick" is an implementation-dependent time interval, usually 1/50 or 1/60 of a second. The actual delay may vary from the requested tick count by plus or minus one tick.

If the specified tick count is zero, then the calling program is suspended only long enough to allow any other ready processes to run (a so-called "courtesy" dispatch).

T-Function 3 **Allocate Memory**

Entry Arguments

Reg	Description
C	= 3
DE	= byte-length of requested segment

Returned Value

Reg	Description
A	= 0 if successful -1 if insufficient memory
HL	= segment address (if successful)

Explanation

The Allocate Memory function allocates a contiguous memory segment of the byte-length requested in register DE. If successful, the starting address of the allocated segment is returned in register HL.

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

T-Function 4 **Deallocate Memory**

Entry Arguments

Reg	Description
C	= 4
DE	= segment address

Explanation

The Deallocate Memory function returns a previously-allocated memory segment to the pool of available memory space.

NOTE: This function is not intended for use by transient programs, and must be used with great care. The address passed in DE must be a segment starting address returned by a prior call to C-function 3 (Allocate Memory), otherwise a system crash may occur.

T-Function 5 **Send Interprocess Message**

Entry Arguments

Reg	Description
C	= 5
DE	= message node address
HL	= message address

Explanation

The Send Interprocess Message function provides a means to send messages from one process to another. Register DE specifies the address of a 10-byte message node which must be initialized as follows:

```
MSGNOD: .WORD 0         ;semaphore count
        .WORD MSGNOD+2  ;semaphore head
        .WORD MSGNOD+2  ;    "       "
        .WORD MSGNOD+4  ;msg chain head
        .WORD MSGNOD+4  ; "    "     "
```

Register HL specifies the address of the message to be sent, which must be prefixed by a 4-byte linkage as follows:

```
MESSAG: .WORD 0         ;message linkage
        .WORD 0         ;    "        "
        .BYTE ...       ;message text
        .BYTE ...       ;(any length)
```

NOTE: This function is not intended for use by transient programs, and must be used with great care.

T-Function 6 **Receive Interprocess Message**

Entry Arguments

Reg	Description
C	= 6
DE	= message node address

Returned Value

Reg	Description
HL	= message address

Explanation

The Receive Interprocess Message function provides a means to receive messages sent by another process using C-function 5 (Send Interprocess Message). Register DE specifies the address of a 10-byte message node which must be initialized as follows:

```
MSGNOD: .WORD 0          ;semaphore count
        .WORD MSGNOD+2   ;semaphore head
        .WORD MSGNOD+2   ;     "      "
        .WORD MSGNOD+4   ;msg chain head
        .WORD MSGNOD+4   ; "    "    "
```

If no message is available from the specified message node, the calling process is suspended until a message arrives. This function returns in HL the address of the received message prefixed by a 4-byte linkage.

NOTE: This function is not intended for use by transient programs, and must be used with great care.

T-Function 7 **Set Error Address**

Entry Arguments

Reg	Description
C	= 7
DE	= error intercept routine address, or 0 to restore default error handling

Explanation

The Set Error Address function enables a program to establish its own error intercept routine to intercept and process unrecoverable disk errors. The address of the intercept routine is passed in register DE. Normal TurboDOS error diagnosis is supressed.

The error intercept routine must not call any TurboDOS functions, and must return via a RET instruction with register A set to the desired error recovery alternative:

A-reg	Recovery Action
0	retry operation
+1	ignore error
-1	abort program

If the Set Error Address function is called with register DE set to zero, normal TurboDOS error diagnosis is restored. This also happens automatically when the program terminates.

T-Function 8 **Set Abort Address**

Entry Arguments

Reg	Description
C	= 8
DE	= abort intercept routine address, or 0 to restore default abort handling

Explanation

The Set Abort Address function enables a program to establish its own abort intercept routine to intercept and process user-requested aborts (in response to attention-requests or disk errors). The address of the intercept routine is passed in register DE.

The abort intercept routine may exit via a RET instruction to resume execution of the program at the point of interruption. Alternatively, it may proceed with any desired wrap-up processing and then terminate the program.

If the Set Abort Address function is called with register DE set to zero, normal TurboDOS abort handling restored. This also happens automatically when the program terminates.

T-Function 9 **Set Date and Time**

Entry Arguments

Reg	Description
C	= 9
HL	= Julian date (0 is 31 December 1947)
D	= hours (0-23, binary integer)
E	= minutes (0-59, binary integer)
B	= seconds (0-59, binary integer)

Explanation

The Set Date and Time function sets the system date and time. The Julian date passed in register HL is the number of days since the base date of 31 December 1947. Dates prior to the base date are represented by negative values.

The system date and time may also be set by means of C-function 104 (Set Date and Time), but the format of arguments is considerably different.

T-Function 10 **Get Date and Time**

Entry Arguments

Reg	Description
C	= 10

Returned Value

Reg	Description
HL	= Julian date (0 is 31 December 1947)
D	= hours (0-23, binary integer)
E	= minutes (0-59, binary integer)
B	= seconds (0-59, binary integer)
A	= system tick count

Explanation

The Get Date and Time function returns the system date and time. The Julian date returned in register HL is the number of days since the base date of 31 December 1947. Dates prior to the base date are represented by negative values.

The system tick count returned in register A is incremented every system tick. It counts from zero to 255, then wraps around to zero. A system tick is an implementation-dependent time interval, usually 1/50 or 1/60 of a second.

The system date and time may also be interrogated by means of C-function 105 (Get Date and Time), but the format of returned values is considerably different.

T-Function 11 **Rebuild Disk Map**

Entry Arguments

Reg	Description
C	= 11
E	= disk drive: 0 for drive A 1 for drive B : 15 for drive P

Returned Value

Reg	Description
A	= 0 if successful -1 if disk write-protected or has files open

Explanation

The Rebuild Disk Map function regenerates the allocation map on the disk drive specified in register E. The principal purpose of this function is to support the FIXMAP command.

T-Function 12 **Return Serial Number**

Entry Arguments

Reg	Description
C	= 12

Returned Value

Reg	Description
HL	= TurboDOS origin number
DE	= TurboDOS unit number
B	= 0 if non-privileged log-on
	80H if privileged log-on
C	= 13H (TurboDOS version 1.3)

Explanation

The Return Serial Number function returns the origin and unit numbers with which this particular copy of TurboDOS was serialized, and may be used in application programs to help prevent unauthorized use.

This function also returns the TurboDOS version number, and a flag which indicates whether or not the current log-on is privileged.

T-Function 13 **Set Compatibility Flags**

Entry Arguments

Reg	Description
C	= 13
E	= compatibility flags: bit 7 = permissive flag bit 6 = suspend flag bit 5 = global-write flag bit 4 = mixed-mode flag bit 3 = logical flag (bits 2-0 not defined)

Explanation

The Set Compatibility Flags function enables a program to modify the rules by which file sharing is done. The meaning of each compatibility flag is described in section 2.

When the program terminates, the compatibility flags revert automatically to the default values assigned to the public symbol COMPAT at system generation.

T-Function 14 **Log-On/Log-Off**

Entry Arguments

Reg	Description
C	= 14
DE	= 0FFFFH (if log-off)
E	= user number 0-31 (if log-on) with bit 7 set for privileged
D	= current disk drive (if log-on): -1 for no change 0 for drive A 1 for drive B : 15 for drive P

Returned Value

Reg	Description
A	= 0 if successful -1 if request invalid

Explanation

The Log-On/Log-Off function is provided to support log-on security via the LOGON and LOGOFF commands. To log-on, this function is called with the desired user number in register E (with bit 7 set if a privileged log-on is desired), and with the desired current drive in register D (or -1 for no change in current drive). To log-off, the function is called with both DE set to 0FFFFH.

After a log-off, another log-on request is not honored until a warm-start or C-function 0 (System Reset) has occurred.

NOTE: When this function is called from a resident system process, the D-register argument is ignored.

T-Function 15 **Load File**

Entry Arguments

Reg	Description
C	= 15
DE	= FCB address

Returned Value

Reg	Description
A	= 0 if successful 1 if not enough memory to load file -1 if file not found

Explanation

The Load File function loads the file specified by the FCB drive, name, and type fields (bytes 0 through 11) into memory starting at the current DMA address. The file need not have been opened. If the specified drive is local to the processor in which the call is made, the program load optimizer is used. If the top of the TPA is reached before the end-of-file is encountered, the loading stops and an error is returned.

This function is used by the TurboDOS command interpreter to load .COM files into the TPA, and may also be used by application programs to fetch overlays.

T-Function 16 Activate Do-File

Entry Arguments

Reg	Description
C	= 16
DE	= FCB address (to activate) 0 (to cancel)

Returned Value

Reg	Description
A	= 0 if successful -1 if file not found

Explanation

The Activate Do-File function causes the file specified by the FCB drive, name, and type fields (bytes 0 through 11) to be activated as a do-file. The file need not have been opened. Any currently-active do-file and/or command line is stacked (to be reactivated when the new do-file has been processed to completion). The principal purpose of this function is to support the DO command.

This function may also be called with DE set to zero to cancel all active and stacked do-files.

T-Function 17 **Disable/Enable Autoload**

Entry Arguments

Reg	Description
C	= 17
E	= 0 to disable autoload -1 to enable autoload

Explanation

The Disable/Enable Autoload function may be used to disable the warm-start autoload feature of TurboDOS, or to re-enable the feature after it has been disabled.

TurboDOS automatically disables the warm-start autoload feature whenever it fails to find the file WARMSTRT.AUT on the current disk during a warm-start. Creating such a file on disk (or changing the current disk to one that contains such a file) will not result in autoloading unless the autoload feature is explicitly re-enabled by means of this function.

T-Function 18 **Send Command Line**

Entry Arguments

Reg	Description
C	= 18
DE	= buffer address (to send) 0 (to cancel)

Explanation

The Send Command Line function allows a program to specify the next command line to be processed by TurboDOS after the program terminates. The buffer address is passed in register DE. The first byte of the buffer must contain the command line byte-length, and the command line text must occupy the second and succeeding bytes of the buffer. Any currently-active command line is stacked, and the new command line is activated.

This function may also be called with DE set to zero to cancel all active and stacked command lines.

T-Function 19 **Return Disk Allocation Information**

Entry Arguments

Reg	Description
C	= 19
E	= disk drive: 0 for drive A 1 for drive B : 15 for drive P

Returned Value

Reg	Description
A	= block size: 3 for 1K blocks 4 for 2K blocks : 7 for 16K blocks plus: bit 7 set if fixed disk bit 6 set if EXM=0 forced
C	= number of blocks in the directory
DE	= number of blocks presently unused
HL	= total number of blocks on the disk

Explanation

The Return Disk Allocation Information function returns various parameters concerning the logical organization of the specified disk drive.

T-Function 20 **Return Physical Disk Information**

Entry Arguments

Reg	Description
C	= 20
E	= disk drive: 0 for drive A 1 for drive B : 15 for drive P

Returned Value

Reg	Description
A	= physical sector size: 0 for 128-byte sectors 1 for 256-byte sectors 2 for 512-byte sectors 3 for 1K sectors : 7 for 16K sectors
BC	= number of reserved (boot) tracks
DE	= total number of tracks on the disk
HL	= number of sectors per track

Explanation

The Return Physical Disk Information function returns various parameters concerning the format and physical organization of the specified disk drive.

T-Function 21 Get/Set Drive Status

Entry Arguments

Reg	Description
C	= 21
E	= disk drive: 0 for drive A 1 for drive B : 15 for drive P
D	= 0 to set the drive read/write 1 to set the drive read-only -1 to return the drive status

Returned Value

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

Explanation

The Get/Set Drive Status function may be used to interrogate the ready and write-protect status of the drive specified by register E. This function may also be used to change the write-protect status of the drive. The code passed in register D controls which of these operations is performed, as indicated above.

T-Function 22 Physical Disk Access

Entry Arguments

Reg	Description
C	= 22
DE	= PDR packet address

Returned Value

Reg	Description
A	= 0 if read/write successful, or drive not ready -1 if read/write unsuccessful, or drive is ready

Explanation

The Physical Disk Access function provides direct access to the physical disk drivers. The principal purpose of this function is to support the BOOT, BACKUP, FORMAT, and VERIFY commands. It is honored for privileged logons only, and may be used only for disk drives local to the calling processor.

Register DE contains the address of a 14-byte physical disk request (PDR) packet with the following structure:

Offset	Description
0	disk operation code (0-4)
1	disk drive (0-15)
2-3	physical track number (base 0)
4-5	physical sector number (base 0)
6-7	number of sectors to read/write
8-9	number of bytes to read/write
10-11	DMA address for read/write
12-13	disk specification table address

The physical operation performed depends upon the disk operation code in the PDR packet.

Explanation (Continued)

If the PDR opcode is 0, the specified number of physical sectors (or bytes) are read from the specified drive, track, and sector into the specified DMA address.

If the PDR opcode is 1, the specified number of physical sectors (or bytes) are written to the specified drive, track, and sector from the specified DMA address.

If the PDR opcode is 2, the type of the specified disk is determined, and an 11-byte disk specification table (DST) is returned at the specified DMA address, structured as follows:

Offset	Description
0	block size (3=1K,4=2K,...,7=16K)
1-2	total number of blocks on disk
3	number of directory blocks
4	sector size (0=128,...,7=16K)
5-6	number of sectors per track
7-8	number of tracks on the disk
9-10	number of reserved (boot) tracks

If the PDR opcode is 3, the ready status of the specified drive is returned in register A (0 if not ready, -1 if ready).

If the PDR opcode is 4, the specified track of the specified drive is formatted, using hardware-dependent formatting information provided at the specified DMA address.

NOTE: Opcodes 0 (read) and 1 (write) require that the PDR packet contain the address of a valid DST for the specified disk. Therefore, opcode 2 (return DST) should be invoked first to obtain the DST.

T-Function 23 **Set Buffer Parameters**

Entry Arguments

Reg	Description
C	= 23
D	= number of buffers (minimum 2)
E	= buffer size: 0 for 128-byte buffers 1 for 256-byte buffers 2 for 512-byte buffers 3 for 1K buffers : 7 for 16K buffers

Explanation

The Set Buffer Parameters function enables the number and size of disk buffers to be changed. The principal purpose of this function is to support the BUFFERS command.

The specified number of buffers must be at least 2. If the specified number of buffers cannot be allocated due to insufficient memory, then TurboDOS allocates as many as it can. The specified buffer size must be as least as large as the largest physical disk sector size being used.

If this function is called from a slave processor without local disk storage, then the function is passed over the network to be processed in the master.

T-Function 24 **Get Buffer Parameters**

Entry Arguments

Reg	Description
C	= 24

Returned Value

Reg	Description
A	= memory size in pages (0 if 64K)
H	= number of buffers
L	= buffer size:
	0 for 128-byte buffers
	1 for 256-byte buffers
	2 for 512-byte buffers
	3 for 1K buffers
	:
	7 for 16K buffers

Explanation

The Get Buffer Parameters function enables the number and size of disk buffers to be interrogated. The principal purpose of this function is to support the BUFFERS command.

If this function is called from a slave processor without local disk storage, then the function is passed over the network to be processed in the master.

T-Function 25 **Lock/Unlock Drive**

Entry Arguments

Reg	Description
C	= 25
E	= disk drive: 0 for drive A 1 for drive B : 15 for drive P
D	= 0 to unlock drive -1 to lock drive

Returned Value

Reg	Description
A	= 0 if successful -1 if drive in-use or already locked by another process

Explanation

The Lock/Unlock Drive function enables a program to secure a lock on a specified disk drive. This function is used by many TurboDOS commands such as BACKUP, CHANGE, FIXDIR, FIXMAP, FORMAT, and VERIFY to ensure that they cannot compromise the processing of other users.

T-Function 26 **Flush/Free Buffers**

Entry Arguments

Reg	Description
C	= 26
E	= disk drive: 0 for drive A 1 for drive B : 15 for drive P
D	= subfunction flags: bit 7 set to free buffers unconditionally bit 6 set to free buffers after disk error abort bit 5 set to continue after disk error abort bit 4 set to return after disk error abort

Explanation

The Flush/Free Buffers function causes all written-to disk buffers for the specified disk drive to be written out (flushed) to the disk. This function may cause disk buffers for the specified drive to be freed, conditionally or unconditionally, according to the subfunction flags passed in register D.

It is suggested that this function be used prior to media changes and physical disk access (T-function 22).

T-Function 27 Get/Set Print Mode

Entry Arguments

Reg	Description
C	= 27
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 mode = 0) queue assignment (if mode = 1) -1 to leave assignment unchanged
B	= spool drive: 0 for drive A 1 for drive B : 15 for drive P -1 to leave spool drive unchanged

Returned Value

Reg	Description
A	= current spool drive
H	= current printer or queue assignment
L	= current print mode

Explanation

The Get/Set Print Mode function is used to set or interrogate print routing, and is provided to support the PRINT command.

Printer and queue assignments are coded thus: 1 for A, 2 for B, ..., 16 for 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 registers B, D, and E are all set to -1, this function simply interrogates and returns the current assignment, mode, and drive.

T-Function 28 **Signal End-of-Print**

Entry Arguments

Reg	Description
C	= 28

Explanation

The Signal End-of-Print function causes an end-of-print condition. If spooling is in effect, the current print file is closed and (if appropriate) enqueued for background printing.

An end-of-print condition may also occur as the result of a warm-start, attention request, or end-of-print character.

T-Function 29 **Get/Set De-Spool Mode**

Entry Arguments

Reg	Description
C	= 29
B	= printer: 0 for printer A 1 for printer B : 15 for printer P
E	= de-spool mode: 0 to process print job 1 to suspend print job 2 to begin print job over 3 to terminate print job -1 to leave mode unchanged
D	= de-spool queue assignment: 0 to set printer off-line 1 for queue A 2 for queue B : 16 for queue P -1 to leave queue unchanged

Returned Value

Reg	Description
A	= 0 if successful -1 if invalid request
H	= current queue assignment (0-16)
L	= current de-spool mode (0 or 1)

Explanation

The Get/Set De-Spool Mode function is used to control background printing, and is provided to support the PRINTER command.

If registers D and E are both set to -1, this function simply interrogates and returns the current queue assignment and de-spool mode for the specified printer.

T-Function 30 **Queue a Print File**

Entry Arguments

Reg	Description
C	= 30
DE	= FCB address
H	= print queue: 0 for queue A 1 for queue B : 15 for queue P
L	= user number (0-31), plus bit 7 set to delete after printing

Returned Value

Reg	Description
A	= 0 if successful -1 if invalid request

Explanation

The Queue a Print File function enqueues a text file on a specified print queue for background printing. The file to be enqueued is identified by the FCB drive, name and type fields (bytes 0 through 11), together with the user number passed in register L.

The drive specified by the FCB must be accessible by the processor in which the specified queue resides, otherwise the request is invalid. To check this, the function may be called with register L set to -1, in which case the FCB drive and requested queue are checked for validity but no file is queued.

T-Function 31 **Flush List Buffer**

Entry Arguments

Reg	Description
C	= 31

Explanation

The Flush List Buffer function is used by TurboDOS during direct printing over the network to force any remaining buffered characters to be printed. There should be no need for an application program to call this function.

T-Function 32 **Network List Out**

Entry Arguments

Reg	Description
C	= 32
E	= output character

Explanation

The Network List Out function is used by TurboDOS during direct printing over the network. There should be no need for an application program to call this function.

T-Function 33 **Remote Console I/O**

Entry Arguments

Reg	Description
C	= 33
E	= console input character, or 0 if no console input available
D	= 0 to detach remote console -1 to attach remote console

Returned Value

Reg	Description
A	= 0 if CONREM not present 1 if successful -1 if executing in master

Explanation

The Remote Console I/O function works in conjunction with the CONREM console driver to support the MASTER command. It passes one byte of console input in register E (if available), and returns a count byte and up to 127 bytes of console output at the current DMA address. There should be no need for an application program to call this function.

T-Function 34 **Get Comm Channel Status**

Entry Arguments

Reg	Description
C	= 34
D	= channel number, plus bit 7 set if remote channel

Returned Value

Reg	Description
A	= 0 if input character not available -1 if input character is available

Explanation

The Get Comm Channel Status function checks to see whether or not an input character is available on the specified comm channel. If a character is available, it returns A = -1. Otherwise, it returns A = 0.

T-Function 35 **Comm Channel Input**

Entry Arguments

Reg	Description
C	= 35
D	= channel number, plus bit 7 set if remote channel

Returned Value

Reg	Description
A	= input character

Explanation

The Comm Channel Input function obtains the next input character from the specified comm channel, and returns in in register A. If no character is available, the calling program is suspended until a character is received.

T-Function 36 **Comm Channel Output**

Entry Arguments

Reg	Description
C	= 36
D	= channel number, plus bit 7 set if remote channel
E	= output character

Explanation

The Comm Channel Output function outputs the character passed in register E on the specified comm channel.

T-Function 37 **Set Comm Baud Rate**

Entry Arguments

Reg	Description
C	= 37
D	= channel number, plus bit 7 set if remote channel
E	= baud rate code (bits 3-0):

0	for 50 baud	8	for 1800 baud
1	for 75 baud	9	for 2000 baud
2	for 110 baud	10	for 2400 baud
3	for 134.5 baud	11	for 3600 baud
4	for 150 baud	12	for 4800 baud
5	for 300 baud	13	for 7200 baud
6	for 600 baud	14	for 9600 baud
7	for 1200 baud	15	for 19200 baud

plus bit 7 set for att'n detection
bit 6 set for CTS handshaking
bit 5 set for input disabled

Explanation

The Set Comm Baud Rate function sets the baud rate and options passed in register E on the specified comm channel.

T-Function 38 Get Comm Baud Rate

Entry Arguments

Reg	Description
C	= 38
D	= channel number, plus bit 7 set if remote channel

Returned Value

Reg	Description
A	= baud rate code (bits 3-0):
	0 for 50 baud, 8 for 1800 baud
	1 for 75 baud, 9 for 2000 baud
	2 for 110 baud, 10 for 2400 baud
	3 for 134.5 baud, 11 for 3600 baud
	4 for 150 baud, 12 for 4800 baud
	5 for 300 baud, 13 for 7200 baud
	6 for 600 baud, 14 for 9600 baud
	7 for 1200 baud, 15 for 19200 baud
	plus bit 7 set for att'n detection
	bit 6 set for CTS handshaking
	bit 5 set for input disabled

Explanation

The Set Comm Baud Rate function interrogates the baud rate and options for the specified comm channel, and returns this information in register A.

T-Function 39 **Set Modem Controls**

Entry Arguments

Reg	Description
C	= 39
D	= channel number, plus bit 7 set if remote channel
E	= modem control vector: bit 7 set for request-to-send, bit 6 set for data-terminal-ready, bits 5-0 unassigned

Explanation

The Set Modem Controls function sets the modem control signals in accordance with the vector passed in register E on the specified comm channel.

T-Function 40 **Get Modem Status**

Entry Arguments

Reg	Description
C	= 40
D	= channel number, plus bit 7 set if remote channel

Returned Value

Reg	Description
A	= modem status vector: bit 7 set for clear-to-send bit 6 set for data-set-ready bit 5 set for data-carrier-detect bit 4 set for ring-indicator bits 3-0 unassigned

Explanation

The Set Modem Status function interrogates the modem status signals for the specified comm channel, and returns this information as a vector in register A.

T-Function 41 **User-Defined Function**

Entry Arguments

Reg	Description
C	= 41
B	= network routing: 0 if always processed locally 1d hex if routed per drive d 2p hex if routed per printer p 3q hex if routed per queue q -1 if routed to default net addr
DE	= user-defined argument passed
HL	= user-defined argument passed

Returned Value

Reg	Description
A	= user-defined value returned
BC	= user-defined value returned
DE	= user-defined value returned
HL	= user-defined value returned

Explanation

The User-Defined Function provides a means for adding user-defined extensions to the operating system taking full advantage of the TurboDOS networking facilities. On entry, register B defines how the request is to be routed over the network. Registers DE and HL plus the 128-byte record at the 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 the USRFCN routine, register BC contains the address of the 128-byte record that was passed. The USRFCN routine may return information to the caller in any of the seven registers A-B-C-D-E-H-L and in the 128-byte record.

T-Function 42 **Reorganize Disk Directory**

Entry Arguments

Reg	Description
C	= 42
E	= disk drive: 0 for drive A 1 for drive B : 15 for drive P

Returned Value

Reg	Description
A	= 0 if successful -1 if disk write-protected or has files open

Explanation

The Reorganize Disk Directory function reorganizes the directory on the disk drive specified in register E. If the hashed-directory flag bit in the volume label has been changed, this function will convert a hashed directory into linear format (or vice versa). The principal purpose of this function is to support the FIXDIR command.

NOTE: In certain cases, this function may take a very long time to complete (possibly hours), and cannot be interrupted once invoked.

T-Function 43 **Select Memory Bank**

Entry Arguments

Reg	Description
C	= 43
E	= 0 to select bank 0 1 to select bank 1 -1 to interrogate current bank

Returned Value

Reg	Description
A	= current bank (0 or 1)

Explanation

The Select Memory Bank function selects the memory bank in which the TPA resides, according to the bank number (0 or 1) passed in register E. If E = -1, this function simply interrogates the current bank mode and returns the current bank number (0 or 1) in register A. This function is honored for privileged log-ons only, and is provided primarily to support the BANK command.

This function has no effect in a non-banked system, and always shows that bank 0 is selected.

MONITOR Command

The MONITOR command provides various facilities under Z80 TurboDOS useful to programmers who have the need to debug or patch programs and files.

Syntax

```
MONITOR
```

Explanation

The MONITOR command operates in an interactive mode. You are prompted by an asterisk * to enter a series of directives from the console. The Q directive (quit) terminates the MONITOR command.

In the directive descriptions that follow, all addresses and other numeric arguments are in hexadecimal, and all file names ("fn") are entered as {d:}filename{.typ}:

Directives

Directive	Explanation
C v1,v2	Calculates the sum and diference of two hex values.
D a1,a2	Dumps the area of memory between the two given addresses in hex. Pause with space, resume with RETURN.
E a	Examines memory starting at the given address. You may substitute a new hex value for each displayed byte of memory. Enter space or RETURN to go on to the next byte, or ESC to exit examine mode.

Directives
(Continued)

Directive	Explanation
F a1,a2{,v{,r}}	Fills the area of memory between the two given addresses with the given value v (or 0 if no value is given). If a repetition factor r is given, the operation is repeated r times (useful with some PROM programmers).
G a	Goes (jumps) to the instruction at the given address.
H	Help menu is displayed, listing all MONITOR directives.
I p	Inputs a byte from I/O port p and displays its hex value.
L fn {a}	Loads the named file into memory starting at the given address (or 0100H if no address is given).
M a1,a2,a3{,r}	Moves the block of memory between a1 and a2 into the area starting at a3. If a repetition factor r is given, the operation is repeated r times (useful with some PROM programmers).
O p,v	Outputs the hex value v to I/O port p.
P a	Puts succeeding typed ASCII data into memory starting at the given address. Terminate by typing CTRL-D (ASCII EOT).

Directives (Continued)

Directive	Explanation
Q	Quits the monitor command.
R al,a2	RAM area between al and a2 is tested, and memory errors are diagnosed. Typing ESC aborts the test early.
S fn {al,a2}	Saves the area of memory between the given addresses in the named file. If no memory addresses are given, then the bounds from the last L (load) directive are used.
T al,a2	Types the area of memory between the given addresses in ASCII. Pause with space, resume with RETURN.
V al,a2,a3	Verifies that the block of memory from al to a2 is identical to the area starting at a3. Any descrepancies are diagnosed.
W vl,...,vN	Scans all of memory for occurrences of the given byte string, and displays Where each occurrence is found.
Y	Displays the highest available address in memory (below TurboDOS and the MONITOR command code).

Examples

```
0A}MONITOR
TurboDOS Monitor -- Copyright etc.
* Y
E7FF
* C E7FF 100
E8FF E6FF
* F 100,E7FF,E5
* D 100,11F
0100 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 etc.
0110 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 etc.
* Q
0A}
```

C =	C-Function Name	Arguments Passed	Values Returned
0	System Reset	-	-
1	Console Input	-	A = char
2	Console Output	E = char	-
3	Raw Console Input	-	A = char
4	Raw Console Output	E = char	-
5	List Output	E = char	-
6	Direct Console I/O	E = -1 (inp/sta)	A = 0/char
		E = -2 (status)	A = 0/-1
		E = -3 (input)	A = char
		E = char (output)	-
7	Get I/O Byte	-	A = I/O byte
8	Set I/O Byte	E = I/O byte	-
9	Print String	DE = &string	-
10	Read Console Buffer	DE = &buffer	-
11	Get Console Status	-	A = 0/-1
12	Return Version	-	H = 0
			L = 31H
13	Reset Disk System	-	-
14	Select Disk	E = drive (0=A)	-
15	Open File	DE = &FCB	A = (-1 if err)
16	Close File	DE = &FCB	A = (-1 if err)
17	Search for First	DE = &FCB	A = (-1 if err)
18	Search for Next	-	A = (-1 if err)
19	Delete File	DE = &FCB	A = (-1 if err)
20	Read Sequential	DE = &FCB	A = (NZ if err)
21	Write Sequential	DE = &FCB	A = (NZ if err)
22	Make File	DE = &FCB	A = (-1 if err)
23	Rename File	DE = &FCB	A = (-1 if err)
24	Return Login Vector	-	HL = vector
25	Return Current Disk	-	A = drive (0=A)
26	Set DMA Address	DE = &DMA	-
27	Get ALV Address	(not supported)	HL = 0
28	Write Protect Disk	-	-
29	Get R/O Vector	-	HL = vector
30	Set File Attributes	DE = &FCB	A = (-1 if err)
31	Get DPB Address	-	HL = &DPB
32	Get/Set User Number	E = -1	A = user number
		E = user number	-
33	Read Random	DE = &FCB	A = (NZ if err)
34	Write Random	DE = &FCB	A = (NZ if err)

C =	C-Function Name	Arguments Passed	Values Returned
35	Compute File Size	DE = &FCB	A = (-1 if err)
36	Set Random Record	DE = &FCB	-
37	Reset Drive	DE = vector	-
40	Write Random 0-Fill	DE = &FCB	A = (NZ if err)
42	Lock Record	DE = &FCB	A = (NZ if err)
43	Unlock Record	DE = &FCB	A = (NZ if err)
46	Get Disk Free Space	E = drive (0=A)	A = 0
47	Chain to Program	(Cmd at 0080H)	-
104	Set Date and Time	DE = &DTP	-
105	Get Date and Time	DE = &DTP	A = seconds/BCD
107	Return Serial Nbr	DE = &SN	-
108	Get/Set Return Code	DE = 0FFFFH	HL = retcode
		DE = retcode	-
110	Get/Set Delimiter	DE = 0FFFFH	A = delimiter
		E = delimiter	-
111	Print Block	DE = &CCB	-
112	List Block	DE = &CCB	-
152	Parse Filename	DE = &PFCB	HL = 0 if EOL -1 if error else &delim

C =	T-Function Name	Arguments Passed	Values Returned
0	Reset O/S	-	-
1	Create Process	DE = &entrypoint HL = &workspace	A = 0/-1
2	Delay Process	DE = tick count	-
3	Allocate Memory	DE = length	A = 0/-1 HL = &memory
4	Deallocate Memory	DE = &memory	-
5	Send I/P Message	DE = &msgnode HL = &message	-
6	Receive I/P Message	DE = &msgnode	HL = &message
7	Set Error Address	DE = 0/&errorcode	-
8	Set Abort Address	DE = 0/&abortcode	-
9	Set Date and Time	HL = Julian date D = hours E = minutes B = seconds	-
10	Get Date and Time	-	HL = Julian Date D = hours E = minutes B = seconds A = tick count
11	Rebuild Disk Map	E = drive (A=0)	A = 0/-1
12	Return Serial Nbr	-	HL = origin # DE = unit # B = 80H if priv C = 13H version
13	Set Compatability	E = compatflags	-
14	Log-On/Log-Off	DE = 0FFFFH (off) D = -1/drive (on) E = user nbr (on)	A = 0/-1
15	Load File	DE = &FCB	A = 0/1/-1
16	Activate Do-File	DE = 0/&FCB	A = 0/-1
17	Dis/Enable Autoload	E = 0 (disable) E = 1 (enable)	-
18	Send Command Line	DE = 0/&buffer	-
19	Get Alloc Info	E = drive (0=A)	A = block size C = dir blocks DE = free blocks HL = tot. blocks

C =	T-Function Name	Arguments Passed	Values Returned
20	Get Physical Info	E = drive (0=A)	A = sector size BC = res. tracks DE = tot. tracks HL = sectors/trk
21	Get/Set Drv Status	E = drive (0=A) D = 0 (set R/W) D = 1 (set R/O) D = -1 (get)	A = 0/-1 L = -1 if ready H = -1 if R/O
22	Phys. Disk Access	DE = &PDR	A = 0/-1
23	Set Buffer Params	D = # of buffers E = buffer size	-
24	Get Buffer Params	-	A = mem. size H = # of buffers L = buffer size
25	Lock/Unlock Drive	E = drive (0=A) D = 0 (unlock) D = -1 (lock)	A = 0/-1
26	Flush/Free Buffers	E = drive (0=A) D = subfunctions	-
27	Get/Set Print Mode	E = print mode D = printer/queue B = spool drive	A = spool drive H = prntr/queue L = print mode
28	Signal End-of-Print	-	-
29	Get/Set Despool Mod	E = despool mode D = queue assgnmt B = printer	A = 0/-1
30	Queue a Print File	DE = &FCB H = print queue L = user#/delete	A = 0/-1
31	Flush List Buffer	-	-
32	Network List Out	E = char	-
33	Remote Console I/O	E = 0/char D = -1 to attach	A = 0/1/-1
34	Get Comm Status	D = channel/rmt	A = 0/-1
35	Comm Channel Input	D = channel/rmt	A = char
36	Comm Channel Output	D = channel/rmt E = char	-

C =	T-Function Name	Arguments Passed	Values Returned
37	Set Comm Baud Rate	D = channel/rmt E = baudrate	-
38	Get Comm Baud Rate	D = channel/rmt	A = baudrate
39	Set Modem Controls	D = channel/rmt E = vector	-
40	Get Modem Status	D = channel/rmt	A = vector
41	User-Defined Fcn	B = net routing DEHL userdef	ABCDEHL userdef
42	Reorg Disk Dir	E = drive (0=A)	A = 0/-1
43	Select Memory Bank	E = -1 (interrog) E = 0 (bank 0) E = 1 (bank 1)	A = bank #

INDEX