DD-DOS

A Small DOS Virtual Machine
By dd86k

Technical Reference Manual

Revision 2018.1, July 2018

Revision History

Revision	DD-DOS V.	Date	Description
v0.0.0	v0.0.0	2017-01-16	Internal Initial Release
v0.1.0	v0.0.0	2017-01-17	Additions
v0.2.0	v0.0.0	2017-01-23	Additions, improvements, and corrections.
v0.2.1	v0.0.0	2017-01-23	Internal Shell and fixes.
v0.3.0	v0.0.0	2017-02-01	Added Sections, debugging additions, file formats.
v0.4.0	v0.0.0	2017-03-01	Additions, small fixes.
v0.5.0	v0.0.0	2017-04-22	First public release: A few additions, a lot of corrections.
v0.5.1	v0.0.0	2017-04-22	Fixed table of content and figures.
v0.6.0	v0.0.0	2017-04-29	Added and re-organized some major sections: X86 Encodings, Micro-processors, Systems, Operating systems, Glossary, and Index sections were added. Modified all legends to label tables as such instead of figures. Many English-related and typographic issues resolved.
v0.6.1	v0.0.0	2017-04-29	Added REG field encoding table, removed one confusion.
v0.6.2	v0.0.0	2017-04-29	Added 32-bit ModR/M table. Fixed a few things including possible confusion in §4.
v0.6.3	v0.0.0	2017-12-28	Modifications (logger, cli, disclaimer, document styles a bit) Changed style
v0.6.4	v0.0.0-0	2018-02-04	Corrections for clarity
v0.6.5	v0.0.0-0	2018-03-26	Changed from Reference Manual to Technical Manual Changed copyright information
v0.7.0	v0.0.0-0	2018-04-08	Added x86 instruction format table Adjusted fonts Redone page numbering Revisited every articles, fixed typos and sentence structure
v0.7.1	v0.0.0-0	2018-04-13	Fixed typos Modified footer style Revisited Glossary
2018.0	v0.0.0-0	2018-04-28	Whole document corrections and adjustments, including footer Add x86 flags

Revision	DD-DOS V.	Date	Description
2018.1	v0.0.0-0	2018-07-07	Revised entire document Add Initialization flow, interrupt descriptions, and critical bug codes, tables (FCB, etc.) Please note that all future document changes will be noted in a separate document. Therefore deprecating the revision log within this document and will be removed next revision of this document.

Table of Content

Index of Tables

Preface

About This Document

This document is about the internal technical details of the DD-DOS (https://github.com/dd86k/dd-dos) project for developers, engineers, enthusiasts, and the curious mind. It was initially written for myself as a memory aid.

This document is written entirely by dd86k (https://github.com/dd86k). It features a collection of information gathered through out on the web and various documentation about the various products DD-DOS is based on, notably the Intel 8086, the IBM PC XT, and the Phoenix BIOS.

Any comments, additions, suggestions, and fixes can be sent via email (devddstuff@gmail.com) or as an Issue on Github with the label “Technical Manual”.

Happy reading!

Trademarks

All products mentioned in this document are trademarked products to the respectful owner.

MS-DOS and Windows are trademarks of Microsoft Corporation.

Disclaimer

The information contained within this document are solely advisory, should not be substituted for professional advice, and may change at any given time without any warning.

Conventions

This document uses various conventions, explained below.

Xh	Hexadecimal-formatted number, usually grouped by two bytes (e.g. FF_FFFFh)
Xb	Binary-formatted number, the b suffix may be optional
`Mono-spaced font`	Code or command example
X:X	Bit range, both the maximum and minimum numbers are inclusive, respectively. Can also be used adjacent to a variable. Example 1: AL[0] points to the first bit of the AL register Example 2: AL[2:0] points to position 0 to 2 (range)
X:Xh	Address location, it's effective (calculated) address is calculated by shifting the former value by 4 bits and adding the latter part.

Table 1: Conventions used

This document uses the little-endian byte order exclusively.

1 Introduction

The Microsoft Disk Operating System, mostly known as MS-DOS, derived from 86-DOS, was the most used DOS-based operating system from its time, making it a key product for Microsoft to grow as a large software company during the time.

Ever since the release of 64-bit Windows operating systems, the 16-bit emulation layer, NTVDM, has been stripped out of the system due to x86-64 based processors not supporting the Virtual 8086 mode while running in LONG mode (64-bit).

1.1 Goal

DD-DOS is an open-source project aiming and hoping to be a light and simple DOS-compatible layer, running alongside the very operating system's terminal via emulation on many platforms and means to eventually replace NTVDM, the WindowsNT Virtual DOS Machine layer.

In the far future, I hope to see DD-DOS:

- Run in resource-limited machines, like a low-end Raspberry Pi;
- Run most MS-DOS applications;
- Have its own virtual, visual Windows 3.11 environment.
- Run games, such as the original Rogue and DOOM.

2 Initialization

By default, DD-DOS enters an interactive session, into its virtual shell, within vDOS. If a file is specified, then DD-DOS runs the specified program until it exists.

2.1 Command Line Interface

Syntax:

```
dd-dos [-vPN] [FILE [FILEARGS]]
dd-dos {-V|--version|-h|--help}
```

Options:

`-V, --version`

Displays the version screen, then exits.

`-h, --help`

Displays the help screen, then exits.

`-v`

Increment verbosity. Verbose levels are explained in Logging.
This setting is not recommended for TUI applications.
Debug builds automatically sets verbose mode at LOG_DEBUG.

`-P`

Disables thread sleeping between instructions, enabling maximum performance.

`-N`

Disables start-up banner logo and start-up messages.

2.2 Initialization flow

From the command line interface, DD-DOS initiates in the following order:

1. Print verbose and informal messages
2. Print "DD-DOS is starting..."
3. Initiate console library
4. Initiate DD-DOS (virtual processor, memory, etc.)
5. Print banner
6. If FILE is mentioned from the CLI
   a. If found, FILE is loaded and executed
   b. Otherwise, an error code is returned and execution stops
7. If FILE is not mentioned
   a. The flow is transferred to vDOS' virtual shell

3 Settings

User settings are saved in a file named "settings.sdl" under the user profile (under ~/.dd-dos or under the current directory) using the Simple Declarative Language specifications (http://sdlang.org/) in UTF-8.

By default, DD-DOS will not create a settings file if missing, and will assume default settings. If fields are missing, DD-DOS will assume default values for those settings.

Settings are currently not implemented.

Name	Type	Default value	Description
row	Number	25	Console height
column	Number	80	Console width
app	String	null	Starting application
date	Number	Current date	Date, see Date and Time
time	Number	Current time	Time, see Date and Time
cwd	String	.	Current working directory, affects vDOS
videomode	Number	-1	Video mode, see DOS Video Modes

Table 2: DD-DOS Settings

3.1 Terminal Size

DD-DOS requires at least 80x25 characters to operate normally. If the terminal window is too small, it will be resized if possible, otherwise it will terminate with an error message.

3.2 Date and Time

The date and time formats are 32-bit hexadecimal numbers expressed in little-endian.

For time, all fields are optional except for hours. For example, 0225_0A06h converts to 6:10:25.02, and 1007h converts to 7:16:00.00.

Offset	Size	Description
0	1	Hours (CH)
1	1	Minutes (CL, Optional)
2	1	Seconds (DH, Optional)
3	1	Milliseconds (DL, Optional)

Table 3: Time Setting Structure

For date, all fields are required. For example, a number 07CB_0203h would convert to 1995/Feb/03.

Offset	Size	Description

0	2	Year (CX)
2	1	Month (DH)
3	1	Day (DL)

Table 4: Date Setting Structure

4 Instruction Format

This section attempts to address some of the ways the x86 machine code is encoded.

Under the 8086, a typical instruction is encoded with the direction bit (D) and the word bit (W).

7	6	5	4	3	2	1	0
						D	W

Table 5: General Opcode Encoding

The D bit is set whenever the destination is specified in the REG (see: ModR/M Byte) field. Otherwise, the register specified in the REG field is used as the source register.

The W bit is set whenever the instruction operates on WORD data (2 bytes). Otherwise, BYTE data.

On the 8086, only the Segment override prefix exists.

Bytes	0, 1	0, 1	0, 1	0, 1
	Instruction Prefix	Address-size Prefix	Operand-size Prefix	Segment override

Table 6: Optional Instruction Prefixes

Bytes	1, 2	0, 1	0, 1	0, 1, 2, 4	0, 1, 2, 4
	Opcode	ModR/M	SIB	Displacement	Immediate

Table 7: General Instruction Format

4.1 ModR/M Byte

The Mode Register/Modifier byte is an instruction modifier. It can set a memory mode and a destination or source register.

The ModR/M byte contains three fields: MOD, REG, and R/M.

ModR/M Byte

7	6	5	4	3	2	1	0
MOD		**REG**			**R/M**		

Table 8: ModR/M Byte Structure

Only the MOD and R/M fields are used when calculating the effective address. The REG field determines the register source or destination, depending on the destination (D) bit.

MOD	Description
00	Memory Mode, no displacement follows
01	Memory Mode, 8-bit displacement follows
10	Memory Mode, 16-bit displacement follows
11	Register Mode (no displacement)

Table 9: MOD Field Encoding

REG	Register	
	Byte (W = 0)	**Word (W = 1)**
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

Table 10: REG Field Encoding

The Dn notation, where n represents bits, denotes a displacement. The displacements are sign-extended.

Effective Address Calculation (Real-Mode)					
R/M	**MOD = 00**	**MOD = 01**	**MOD = 10**	**MOD = 11**	**MOD = 11 + W**
000	SI + BX	SI + BX + D8	SI + BX + D16	AL	AX
001	DI + BX	DI + BX + D8	DI + BX + D16	CL	CX
010	SI + BP	SI + BP + D8	SI + BP + D16	DL	DX
011	DI + BP	DI + BP + D8	DI + BP + D16	BL	BX
100	SI	SI + D8	SI + D16	AH	SP
101	DI	DI + D8	DI + D16	CH	BP
110	D16	BP + D8	BP + D16	DH	SI
111	BX	BX + D8	BX + D16	BH	DI

Table 11: ModR/M Effective Address Calculation (16-bit)

The ModR/M byte’s effect differ on 80386’s EXTENDED mode.

Effective Address Calculation (Extended Mode)					
R/M	**MOD = 00**	**MOD = 01**	**MOD = 10**	**MOD = 11**	**MOD = 11 + W**
000	EAX	EAX + D8	EAX + D32	AL/AX[1]	EAX
001	ECX	ECX + D8	ECX + D32	CL/CX	ECX
010	EDX	EDX + D8	EDX + D32	DL/DX	EDX
011	EBX	EBX + D8	EBX + D32	BL/BX	EBX
100	SIB Mode	SIB Mode + D8	SIB Mode + D32	AH/SP	ESP
101	D32	EBP + D8	EBP + D32	CH/BP	EBP
110	ESI	ESI + D8	ESI + D32	DH/SI	ESI
111	EDI	EDI + D8	EDI + D32	BH/DI	EDI

Table 12: ModR/M Effective Address Calculation (32-bit)

4.2 SIB Byte

The SIB byte (Scaled Index Base), first used in the 80386, is used for scaled indexed addressing.

SIB Byte

7	6	5	4	3	2	1	0
Scale		**Index**			**Base**		

Table 13: SIB Byte Structure

The SIB byte is similar to the ModR/M byte, but instead of a MOD (memory mode) field, the SIB byte has a Scale (memory scale) field.

SIB Byte					
Scale		**Index**		**Base**	
Value	**Index * Value**	**Index**	**Register**	**Base**	**Register**
00	Index * 1	**000**	EAX	**000**	EAX
01	Index * 2	**001**	ECX	**001**	ECX
10	Index * 4	**010**	EDX	**010**	EDX
11	Index * 8	**011**	EBX	**011**	EBX
		100	--	**100**	ESP
		101	EBP	**101**	See note[2]
		110	ESI	**110**	ESI
		111	EDI	**111**	EDI

1 The WORD bit in the opcode specifies 8 or 32-bit register size. The OPERAND-SIZE byte prefix selects an 16-bit register.
2 Displacement (D32) only if MOD is cleared, otherwise EBP if MOD is set to 01 or 10.

Table 14: SIB Byte Fields

The SIB byte heavily depends on the ModR/M byte for its effective address calculations. The immediate Displacement (D) follows the SIB byte. The Base (B) field can be used in any memory modes (MOD), except when Base is set to b101.

Effective Address Calculation				
Index	**MOD = 00**	**MOD = 00 (B=101)**	**MOD = 01**	**MOD = 10**
000	B32 + EAX*S	EAX*S + D32	B8 + EAX*S + D32	B32 + EAX*S + D32
001	B32 + ECX*S	ECX*S + D32	B8 + ECX*S + D32	B32 + ECX*S + D32
010	B32 + EDX*S	EDX*S + D32	B8 + EDX*S + D32	B32 + EDX*S + D32
011	B32 + EBX*S	EBX*S + D32	B8 + EBX*S + D32	B32 + EBX*S + D32
100	B32	D32	B32 + D8	B32 + D32
101	B32 + EBP*S	EBP*S + D32	B8 + EBP*S + D32	B32 + EBP*S + D32
110	B32 + ESI*S	ESI*S + D32	B8 + ESI*S + D32	B32 + ESI*S + D32
111	B32 + EDI*S	EDI*S + D32	B8 + EDI*S + D32	B32 + EDI*S + D32

Table 15: SIB Byte Effective Address Calculation

5 Micro-processor Emulation

The only supported micro-processor at the current time is the Intel 8086.

5.1 Intel 8086

The Intel 8086 is a 16-bit micro-processor released in 1978. It was a successor of the Intel 8085 and the Intel 8080 microprocessors.

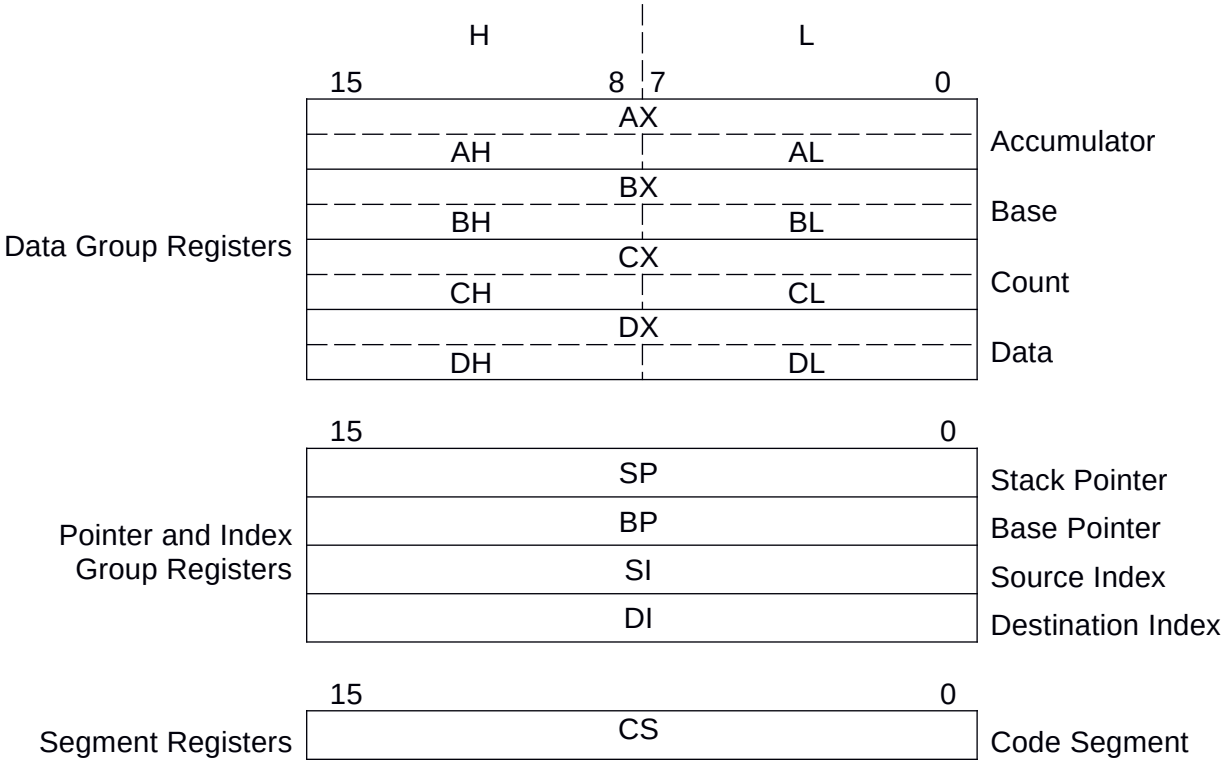
The external address bus is 20-bit wide, providing 1 MB (10_0000h) of physical address space. A new segmented memory model is introduced, making memory accesses into higher memory easier, which provides more than the conventional 640 KB via the segments registers. However, a linear address space is limited to 64 KB due to the 16-bit addressing and index registers.

Models are known to run between 5 and 10 MHz (2 hecto-nanoseconds or 1 hecto-nanosecond per cycle respectively plus an average of 40 nanoseconds of latency for an instruction).

5.1.1 Registers

The Intel 8086 is equipped with 14 16-bit registers:

- 4 general 16-bit registers: AX, BX, CX, DX;
- 4 Index registers: SI, DI, BP, SP;
- 4 segment registers: CS, DS, ES, and SS;
- 1 Program Counter: IP; And 1 Flag register.



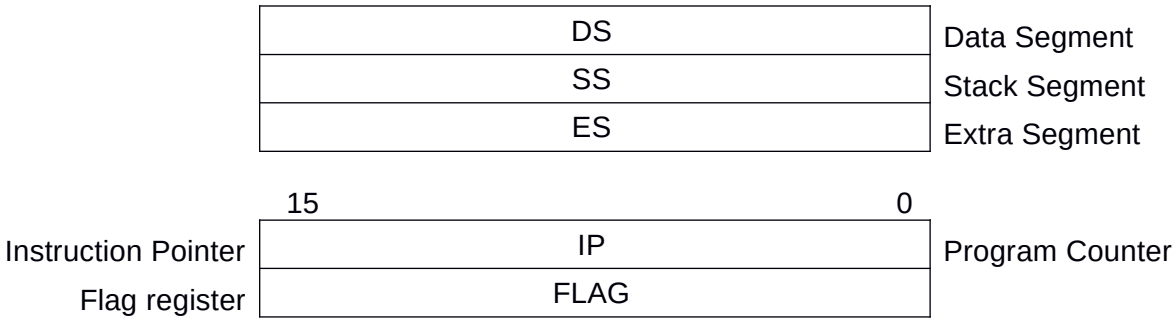


Table 16: Intel 8086 Registers

5.1.2 Flags

The Intel 8086 comes equipped with 9 flags. Every flag has their own position in the FLAG register.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				OF	DF	IF	TF	SF	ZF		AF		PF		CF

Table 17: Intel 8086 Flags

The following sections describe the flags starting at the furthest bit at the right (at bit 0).

5.1.2.1 CARRY FLAG

At bit 0, when set, the CF flag indicates the high-order bit is set. That is, whenever the bit carries pat over to the destination operand; cleared otherwise.

For example, the decimal number 256 is represented as 0b1_0000_0000 in binary, thus the bit got carried to a new nibble, since a BYTE number can only hold up to 8 bits of information.

5.1.2.2 PARITY FLAG

At bit 2, when set, the PF flag indicates the low-order bits has an even number of set bits.

5.1.2.3 AUXILIARY FLAG

At bit 4, the AF flag is set if the low-order overflowed. It is primarely used with BCD arithmetics. Similar to the Carry Flag.

5.1.2.4 ZERO FLAG

At bit 6, when set, the ZF flag indicates the result is zero.

5.1.2.5 SIGN FLAG

At bit 7, when set, the SF flag indicates the sign bit, at the most left, used for signed decimal arithmetic, is set.

5.1.2.6 TRAP FLAG

At bit 8, when set, the TF flag indicates single-stepping for debugging is enabled. This flag is cleared when processing an interrupt.

5.1.2.7 INTERRUPT FLAG

At bit 9, the IF flag is set if the processor responds to maskable interrupt requests. This flag is cleared when processing an interrupt.

5.1.2.8 DIRECTION FLAG

At bit 10, the DF flag controls the flow of string instructions.

5.1.2.9 OVERFLOW FLAG

At bit 11, the OF flag is set if the result overflowed either positively or negatively.

5.1.3 Reserved Memory Locations and I/O Ports

The Intel 8086 reserves some memory for I/O mapping and functions, leaving 1,048,432 bytes (1023 KB, FFF70h) of memory and 65,528 (FFF8h) I/O ports for user-applications.

Memory Locations		I/O Locations	
Reserved	F_FFFCh		
Dedicated	F_FFF0h	Open	100h
Open	80h		
Reserved	14h	Reserved	F8h
Dedicated	0	Open	0

Table 18: Intel 8086 Reserved Memory and I/O Ports

5.2 Intel i486 DX

In 1982, the Intel 80286 introduced the protected mode and unsigned arithmetic operations.

In 1986, the Intel 80386, mostly known as the i386 or 386 DX, introduced the 32-bit extension of the 80286 architecture alongside 32-bit registers.

In 1989, the Intel 80486, mostly known as the i486 DX, introduced the Floating-Point Unit, the level 1 cache (L1), and the CPUID instruction.

Intel i486 DX emulation support is planned.

6 System Emulation

DD-DOS aims to emulate the IBM PC XT as close as possible.

6.1 IBM PC XT

Equipped with an Intel 8088, a variant of the Intel 8086 with an 8-bit communication bus, the original IBM PC was a success back in 1981, making it the bare-bone for future IBM PC compatible computers. It included PC-DOS, a variant of MS-DOS, with a memory bank that ranged from 16 KB to 256 KB of RAM.

The IBM PC XT variant, also known as the IBM 5160, released in 1983, included a hard-drive, and memory options ranging from 128 KB to 640 KB.

This section refers to the IBM 5160.

6.1.1 Memory Map

The following memory map corresponds to the IBM 5160.

Address		Function
Decimal	**Hexadecimal**	
1024 KB 960 KB	10_0000h F_0000h	F_E000 – BIOS location BIOS and BASIC 64 KB Base System ROM
   768 KB	 C_0000h	192 KB ROM Expansion and Control C_8000h – Fixed Disk Control
   640 KB	 A_0000h	128 KB of reserved memory B_8000h – Color graphics display B_0000h – Monochrome display
256 KB	4_0000h	384 KB of memory expansion
0	0	256 KB of Read/Write on-board memory

Table 19: IBM PC XT Memory Map

6.1.2 I/O Port Access

The IBM 5160 Technical Reference Manual defines the following I/O port ranges which are accessible via the IN and OUT instructions.

Range	Description
000h-00Fh	DMA Chip (8237A-5)
020h-021h	Interrupt 8259A
040h-043h	Timer 8253-5

Range	Description
060h-063h	PPI 8255A-5
080h-083h	DMA Page Registers
0A0h[1]	Non-Maskable Interrupt Mask Register
0C0h	Reserved
0E0h	Reserved
200h-20Fh	Game Control
210h-217h	Expansion Unit
220h-24Fh	Reserved
278h-27Fh	Reserved
2F0h-2F7h	Reserved
2F8h-2FFh	Asynchronous Communications (Secondary)
300h-31Fh	Prototype Card
320h-32Fh	Fixed Disk
378h-37Fh	Printer
380h-38Ch[2]	Synchronous Data Link Control (SDLC) Communications
380h-389h	Binary Synchronous Communications (Secondary)
3A0h-3A9h	Binary Synchronous Communications (Primary)
3B0h-3BFh	IBM Monochrome Display/Printer
3C0h-3CFh	Reserved
3D0h-3DFh	Color/Graphics
3E0h-3E7h	Reserved
3F0h-3F7h	Diskette
3F8h-3FFh	Asynchronous Communications (Primary)

Table 20: IBM PC XT I/O Address Map

1 At power-on, the NMI into the 8088 is masked off.
To set the mask on: Send 80h. To unset: Send 00h.

2 SDLC Communications and Secondary Binary Synchronous Communications cannot be used together because their addresses overlap.

7 Operating System Emulation

DD-DOS aims to emulate MS-DOS 5.0 as close as possible.

7.1 Memory Map

MS-DOS 5.0 follows this memory which DD-DOS SHOULD follow as well.

Command transient	High memory
Program memory	
Command resident	
MS-DOS	700h, System Data and System Device Drivers
I/O system	500h, DOS Communication Area 400h, ROM Communication Area
Interrupt vectors	0h

Table 21: MS-DOS Memory Map

The MEM /DEBUG command SHOULD provide more information about memory usage.

7.2 File Attributes

When possible, file attributes are translated between the host operating system and DD-DOS.

Bit	Description
0	Read-only
1	Hidden
2	System
3	Volume label (unused)
4	Sub-directory
5	Archive

Table 22: MS-DOS File Attributes

Bits	Description
4:0	Seconds, 2 seconds increments
10:5	Minutes

Bits	Description
15:11	Hours

Table 23: MS-DOS Directory Time Format

Bits	Description
4:0	Day
8:5	Month
15:9	Year (+ 1980)

Table 24: MS-DOS Directory Date Format

8 Executable Formats

DD-DOS currently supports COM and MZ executable images.

Format	Current Status
COM	Mostly implemented; Needs tuning
MZ	Mostly implemented; Needs tuning
NE	Unsupported
LE	Unsupported; Not planned
LX	Unsupported; Not planned

Table 25: DD-DOS Executable Support

8.1 Program Segment Prefix

The Program Segment Prefix (PSP) is a 256-byte process table (in memory) under MS-DOS systems for the COM and MZ executable images. It contains information about the running application. The PSP will be cleared alongside the application when terminated, unless INT 27h or INT 21h/AH=31h is specified (terminate and stay resident).

The PSP table is loaded into memory before the main application (in low memory).

Offset	Size	Description
0	2	CP/M Exit (INT 20h) (machine code)
2	2	First segment location
4	1	Reversed
5	5	Far call to CP/M compatibility mode within DOS (machine code)
10	4	Previous programs terminate address (INT 22h)
14	4	Previous programs break address (INT 23h)
18	4	Previous programs critical address (INT 24h)
22	2	Parent's PSP segment (usually COMMAND.COM internal)
24	20	Job File Table (used for file redirection, internal)
44	2	Environment segment
46	4	Entry to call INT 21h (SS:SP) (internal)
50	2	JFT size (internal)
54	4	Pointer to the JFT (internal, see Job File Table)
58	4	Pointer to the previous PSP (only used by SHARE in DOS 3.3 and later)
62	4	Reserved
64	2	DOS version

Offset	Size	Description
66	14	Reserved
80	3	DOS far call (INT 21h and RETF) (machine code, CDh, 21h, CBh)
83	2	Reserved
85	7	Reserved (Or can be used for extending FCB 1)
92	16	Unopened Standard FCB 1 (see File Control Block)
108	20	Unopened Standard FCB 2 (overwritten if FCB 1 is opened)
128	1	Number of bytes on the command-line
129	127	Command-line, terminates with CR character (Dh)

Table 26: PSP Table Structure

8.1.1 Job File Table

The JFT, known as the Job File Table, is a DOS data structure containing file handles. Whenever a file is opened, the function (may it be `open`, or `_open`) returns an index from the JFT that points to the entry in the System File Table (SFT).

When a program starts, the first five entries are pre-connected to the standard devices.

Handle	Description
0	Standard input (stdin)
1	Standard output (stdout)
2	Standard error (stderr)
3	Standard auxiliary stream (stdaux)
4	Standard printer stream (stdprn)

Table 27: DOS Standard Handles

8.1.2 File Control Block

The FCB, known as the File Control Block, is a file system structure where the state of an open file is held managed by the operating system, but kept in the program memory, and was used until the introduction of the FAT32 filesystem.

Offset	Size	Description
0	1	Drive number (0=Default, 1=A:, etc.) FFh signals an extended FCB (see Extended FCB)
1	8	Space-padded file name
9	3	Space-padded file extension
12	2	Current block number

Offset	Size	Description
14	2	Logical record size
16	4	File size
20	2	Date of last write (see TODO)
22	2	Time of last write (see TODO)
24	8	Reserved
32	1	Record within current block
33	4	Random access record number, if record size is bigger than 64 bytes, high byte is omitted

Table 28: File Control Block Structure

The driver number has the extended FCB signature, and the defines extra fields and the extended FCB structure defines additional fields to the standard FCB structure.

Offset	Size	Description
0	1	Extended FCB signature, FFh
1	5	Reserved
6	1	File attribute
7	36	Standard FCB structure

Table 29: Extended File Control Block Structure

8.1.3 Data Transfer Area

The Disk Transfer Area, known as the DTA, is an internal structure used by MS-DOS for I/O operations, mostly for searching filenames. Most of these fields are undocumented. DD-DOS may spawn a DTA if needed for compatibility. It's usually located at PSP:80h when a program starts, and may be relocated using INT 21h/AH=1Ah.

Offset	Size	Description
0	1	Attributes used in searching
1	1	Drive used in search
2	11	Search file name used
13	2	Directory entry number (0-based)
15	2	Starting cluster number for current directory, 0 for root directory
17	2	Reserved
19	2	Starting cluster number for current directory, 0 for root directory

Offset	Size	Description
21	1	Attributes for matching file
22	2	File time stamp
24	2	File date stamp
26	2	File size
28	13	ASCIIZ filename and extension, space-padded

Table 30: DOS Data Transfer Area Structure

8.1.4 System File Table

A system file table is a structure used internally within MS-DOS. The SFT contains a FCB table at the end. DD-DOS uses the MS-DOS 4.0 format.

Offset	Size	Description
0	4	Pointer to the next system file table; An offset of FFFFh indicates this is the last table within the chain
4	2	Number of file descriptors in the table
6	Variable	FCB table, 59 bytes on a local file, 61 bytes on network file

Table 31: DOS System File Table Structure

8.1.5 FCB Table

The FCB table differs from a normal FCB. DD-DOS uses the MS-DOS 4.0 format.

Offset	Size	Description
0	2	Number of handles referring to this file; Zero if file is no longer opened
2	2	Open mode; If bit 15 is set, file is opened via FCB
4	1	File attributes (see DOS File Attributes)
5	2	Device information (see Character Device Attributes and Block Device Attributes)
7	4	(Character device) Pointer to device driver header (see Device Driver Header Structure) (Block device) Pointer to DOS Device Control Block, or REDIR data
11	2	Starting file cluster
13	2	File time (see DOS Directory Time Attributes)
15	2	File date (see DOS Directory Date Attributes)
17	4	File size in bytes

Offset	Size	Description
21	4	Current file position
25	2	Relative cluster within file of last cluster read; If this is a network redirection, this is a pointer to a REDIRIFS record and this field is 4 bytes
27	2	(Local file) Sector number of directory
29	4	(Local file) Sector entry number of directory
33	1	(Local file) Directory entry number within sector
34	11	Filename (FCB format)
45	4	(SHARE) Pointer to previous SFT sharing the same name
47	2	(SHARE) Number of network machine opening the file
49	2	PSP segment of file owner
51	2	(SHARE) Offset within SHARE code segment of sharing record; Zero for none
53	2	Absolute cluster number of last cluster read; Zero if file never been read/written
55	4	Pointer to file IFS driver; Zero if native DOS

Table 32: DOS FCB Table Structure

8.2 COM Files

COM files are extremely simple raw binary files and have no header data. They cannot exceed a linear memory segment (64KB – PSP, FF00h) in size.

To load a COM file (in MS-DOS) in memory, the DS and ES registers must be pointing at the start of the file with an origin (location counter) of 100h (PSP size), then jump to the start with an offset of 100h (CS:0100).

8.3 MZ Files

Introduced in MS-DOS 2.0, MZ files are relocatable executable files running under real-address mode.

Offset	Size	Field	Description
0	2	e_magic	“MZ” (5A4Dh) signature
2	2	e_cblp	Number of bytes on the last page of the file
4	2	e_cp	Number of pages in the file
6	2	e_crlc	Number of entries in the relocation table
8	2	e_cparhdr	Number of paragraphs[1] in the header

1 A paragraph is 16 bytes in size.

Offset	Size	Field	Description
10	2	e_minalloc	Number of paragraphs required by the program excluding the PSP and the program image
12	2	e_maxalloc	Number of paragraphs requested by the program
14	2	e_ss	Relocatable segment address for the Stack Segment (SS) register
16	2	e_sp	Initial Stack Pointer (SP) value
18	2	e_csum	The sum of all other fields should be zero
20	2	e_ip	Initial Instruction Pointer (IP) value
22	2	e_cs	Relocatable segment address for the Code Segment (CS)
24	2	e_lfarlc	Absolute offset to the relocation table
26	2	e_ovno	Overlay management value. If zero, this is the main executable
28	32	e_res	Extra information for the main program's overlay management
60	4	e_lfanew	Location of the new header, if present

Table 33: MZ Header Structure

If the e_minalloc and e_maxalloc fields are cleared, the program is loaded in high memory. Otherwise, the program is loaded in low memory, above the PSP structure.

To load a MZ file, the code section is copied from the executable to memory. The relocation table is read and far pointers are adjusted in memory. The registers are set up with the following:

- AL and AH for the drive letter status;
- DS:ES that points to the PSP segment;
- SS:SP that points to the stack pointer.

The data after the header lies the relocation table. To get the position of the relocation within the file, it requires computing the physical address from the segment-offset pair. Note that the raw binary code starts on a paragraph boundary within the executable file. All segments are relative to the start of the executable in memory, and this value must be added to every segment if relocation is done manually.

Then the processor jumps to CS:IP (typically CS:0100), after setting up the PSP.

8.4 NE Files

The New Executable format was introduced in MS-DOS 4.0 and Windows 1.01. It keeps the MZ stub. The header starts where `e_lfanew` points to, and complies with the following structure:

Offset	Size	Field	Description
0	2	ne_magic	"NE" (454Eh) signature
2	1	ne_ver	Version number

Offset	Size	Field	Description
3	1	ne_rev	Revision number
4	2	ne_enttab	Offset of Entry Table
6	2	ne_cbenttab	Number of bytes in Entry Table
8	4	ne_crc	CRC32 checksum of the whole file
12	2	ne_flags	Flag word
14	2	ne_autodata	Automatic data segment number
16	2	ne_heap	Initial heap allocation
18	2	ne_stack	Initial stack allocation
20	4	ne_csip	Initial CS:IP setting
24	4	ne_sssp	Initial SS:SP setting
28	2	ne_cseg	Count of file segments
30	2	ne_cmod	Entries in Module Reference Table
32	2	ne_cbnrestab	Size of non-resident name table
34	2	ne_segtab	Offset of Segment Table
36	2	ne_rsrctab	Offset of Resource Table
38	2	ne_restab	Offset of resident name table
40	2	ne_modtab	Offset of Module Reference Table
42	2	ne_imptab	Offset of Imported Names Table
44	4	ne_nrestab	Offset of Non-resident Names Table
48	2	ne_cmovent	Count of movable entries
50	2	ne_align	Segment alignment shift count
52	2	ne_cres	Count of resource segments
54	2	ne_psegcsum	Offset to segment checksums
56	2	ne_pretthunks	Offset to return thunks
58	2	ne_psegrefbytes	Offset to segment ref. bytes
60	2	ne_swaparea	Minimum code swap area size
62	2	ne_expver	Expected Windows version number

Table 34: NE Header Structure

8.5 LE and LX Files

Both formats were introduced in OS/2 2.0.

The LX format supports 32-bit segments and was used under OS/2 and the Watcom Compiler/Extender (DOS).

The LE format supports mixed 16, 32, and 64-bit segments and was used under Windows 3.x, OS/2, and Windows 9x, mostly as VxD device drivers.

Headers start where e_lfanew (from the MZ structure) points to.

Offset	Size	Field	Description
0	2	e32_magic	“LX” (584Ch) or “LE” (454Ch) signature
2	1	e32_border	Byte order
3	1	e32_worder	Word order
4	4	e32_level	Linear EXE Format Level. Increments from 0
8	2	e32_cpu	Target processor i286 = 1 i386 = 2 i486 = 3
10	2	e32_os	Target operating system OS/2 = 1 Windows = 2 DOS 4.x = 3 Windows386 = 4
12	4	e32_ver	Module version
16	4	e32_mflags	Module flags
20	4	e32_mpages	Number of pages in module
24	4	e32_startobj	The Object number to which the Entry Address is relative
28	4	e32_eip	Entry Address of module
32	4	e32_stackobj	The Object number to which the ESP is relative
36	4	e32_esp	Starting stack address of module
40	4	e32_pagesize	The size of one page for this system
44	4	e32_pageshift	The shift left bits for page offsets
48	4	e32_fixupsize	Total size of the fixup information in bytes
52	4	e32_fixupsum	Checksum for fixup information
56	4	e32_ldrsize	Size of memory resident tables
60	4	e32_ldrsum	Checksum for loader section
64	4	e32_objtab	Object table offset
68	4	e32_objcnt	Object table count
72	4	e32_objmap	Object page table offset
76	4	e32_itermap	Object Iterated Pages offset
80	4	e32_rsrctab	Resource Table offset

Offset	Size	Field	Description
84	4	e32_rsrccnt	Number of entries in Resource Table
88	4	e32_restab	Resident Name Table offset
92	4	e32_enttab	Entry Table offset
96	4	e32_dirtab	Module Format Directives Table offset
100	4	e32_dircnt	Number of Module Format Directives in the Table
104	4	e32_fpagetab	Fixup Page Table offset
108	4	e32_frectab	Fixup Record Table Offset
112	4	e32_impmod	Import Module Name Table offset
116	4	e32_impmodcnt	The number of entries in the Import Module Name Table
120	4	e32_impproc	Import Procedure Name Table offset
124	4	e32_pagesum	Per-Page Checksum Table offset
128	4	e32_datapage	Data Pages Offset
132	4	e32_preload	Number of Preload pages for this module
136	4	e32_nrestab	Non-Resident Name Table offset
140	4	e32_cbnrestab	Number of bytes in the Non-resident name table
144	4	e32_nressum	Non-Resident Name Table Checksum
148	4	e32_autodata	The Auto Data Segment Object number
152	4	e32_debuginfo	Debug Information offset
156	4	e32_debuglen	Debug Information length
160	4	e32_instpreload	Instance pages in preload section
164	4	e32_instdemand	Instance pages in demand section
168	4	e32_heapsize	Heap size added to the Auto DS Object (16-bit modules only)
172	4	e32_stacksize	Size of stack (undocumented)
176	196	e32_res3	Pad structure to 196 bytes (undocumented)

Table 35: LE/LX Header Structure

9 vDOS Runtime

vDOS is an emulated DOS operating system for dd-dos, acting as the layer between the x86 interpreter and the host system. It includes a virtual shell and some tools for debugging purposes.

By default, vDOS will use `DOS_MAJOR_VERSION` (5) and `DOS_MINOR_VERSION` (0) for its DOS version, as vDOS SHOULD emulate MS-DOS 5.0.

9.1 Internal Shell

The internal shell commands SHOULD resemble the `COMMAND.COM` commands. The commands SHOULD be case-insensitive.

9.1.1 Commands

`DIR`

: View the content of a directory.

`CD`

: Display or change the current directory.

`CLS`

: Clear screen.

`COPY`

: Copy files.

`DEL`

: Delete files.

`EXIT`

: Exit shell.

`MD`

: Make a directory.

`RD`

: Remove a directory.

`REN`

: Rename one or more files.

`TREE`

: Graphically displays the directory structure.

`TYPE`

: Display the content of a text file.

`VER`

: View the DOS version.

9.1.2 Debugging Commands

The debugging commands have the ? character prefix to avoid conflicts with other shell commands.

`??`

Show debugging commands.

`?diag`

Show diagnostic runtime information. Useful for error reporting and debugging.

`?load FILE`

Manually load executable FILE.

`?p`

Toggle vCPU thread sleeping. Enabled by default.

`?panic`

Manually triggers an oops. May terminate DD-DOS.

`?r`

Prints the states of the virtual registers.

`?run`

Manually start the vCPU at CS:IP or EIP. Useful after ?load.

`?s [MEMORY=SS:SP]`

Prints the current stack. Not implemented.

By default, starting from SS:SP.

Example:

```
FF10:0000 A8 02
FF10:0002 42
FF10:0003 90
```

`?u [INSTRUCTIONS=10]`

Disassemble memory in Intel-styled mnemonics. Not implemented.

By default, it will only disassemble 10 instructions starting from CS:IP.

Example:

```
0000FF38 A8 02  TEST 2h
0000FF40 42     INC DX
0000FF41 90     NOP
```

`?v`

Toggle verbose mode.

9.2 Interrupt Handling

The interpreter must reproduce the interrupt procedure and have access to the host operating system services.

The virtual processor handle two types of interruptions:

- Hardware interrupts;
- Software interrupts.

Any kind of interrupt handling is strictly done via vDOS, which makes it the *layer* part of the emulator.

Interrupts fall into three types:

- Faults – An exception that generally can be corrected, flow can continue (LOG_WARNING);
- Traps – An exception that is reported immediately, flow can continue (LOG_INFO);
- Aborts – Severe exception that broke the program or task flow and could not be recovered (LOG_ERROR).

Please note that only the most essential hardware, BIOS, and DOS interrupts are listed below.

9.3 Hardware Interrupts

Hardware-triggered interrupts may origin from the virtual CPU, emulated external hardware, or the host operating system.

9.3.1 INT 00h – Divide Error Exception (#DE)

Type: Fault

A Divide Error occurs when the divisor operand of a division is 0, or the result cannot be represented within the bits in the destination operand.

9.3.2 INT 01h – Debug Exception (#DB)

Type: Trap

The Debug Exception is used for single stepping debugging. This interrupt is triggered if the TF flag is set.

9.3.3 INT 02h – Non-Maskable Interrupt

Type: Not applicable

The NMI is triggered on non-recoverable hardware errors. Some NMI can be masked off (disabled).

9.3.4 INT 03h – Breakpoint Exception (#BP)

Type: Trap

The breakpoint exception is usually triggered for debugging purposes and will call the debugger if possible.

9.3.5 INT 04h – Overflow Exception (#OF)

Type: Trap

Introduced: 80186

Triggered when the INTO instruction has been executed, which checks the OF flag and if set, the overflow trap is generated. The benefit of using the INTO instruction is that if the overflow exception is detected, an exception handler can be called automatically to handle the overflow condition.

9.3.6 INT 05h – Print Screen (IBM PC)

Type: Unknown

Dump the current screen (text) to the first printer available.

Typically invoked by INT 09h when PrintScreen is pressed, but may also be manually invoked. The default interrupt handler is located at F000h:FF54h in the IBM PC. The handler must check if a Bound Range Exception was triggered, otherwise an infinite loop is caused between INT 09h and INT 05h.

9.3.7 INT 05h – Bound Range Exception (#BR)

Type: Fault

Introduced: 80186

Interrupt 05h checks the index of an array. If the index is out of range, the instruction triggers the Bound Range Exception.

9.3.8 INT 06h – Invalid Opcode (#UD)

Type: Fault

Introduced: 80186

Attempting to execute a reserved or undefined operation code, or using an operand type that is invalid will generated an Invalid Opcode Exception.

9.3.9 INT 07h – Device Not Available Exception (#NM)

Type: Fault

Introduced: 80286

Triggered if a coprocessor instruction is meet and no coprocessor were found.

9.3.10 INT 08h – IRQ0 System Timer (IBM PC)

Type: Unknown

Generates a signal around 18.2 times per second, every 54.92ms, by the 8254 timer at channel 0, typically used to keep the clock updated. The usually handler is located at F000h:FEA5h and can be masked via I/O port 21h by sending bit 0.

9.3.11 INT 08h – Double Fault Exception (#DF)

Type: Abort

Introduced: 80286

Generated if the processor detected a second exception while calling an exception handler for a prior exception, typically due to being incapable of processing two handlers serially.

9.3.12 INT 09h – IRQ1 Keyboard Event (IBM PC)

Type: Unknown

Generated whenever a key has been pressed down or is being released.

Typically, the default interrupt handler is located at F000h:E987h and can be masked off with bit 1 on I/O port 21h.

9.3.13 INT 09h – Coprocessor Segment Overrun

Type: Abort (80286, 80386)

Introduced: 80286

Indicates that an 80386 CPU-based systems with an Intel 387 math coprocessor detected a page or segment violation while transferring the middle portion of an Intel 387 math coprocessor operand.

The 80486 and later processors execute INT 0Dh instead, a general protection exception (#GP).

9.3.14 INT 0Ah – Invalid TSS Exception (#TS)

Type: Fault

Introduced: 80286

An error, related to the TSS, occurred, typically triggered when failing to switch a task, or instructions used to retrieve information from the TSS failed.

9.3.15 INT 0Bh – Serial Communications via IRQ3 (COM2)

Type: Unknown

Triggered if COM2 requires attention, that is, if the UART has been programmed to generate interrupts. May be masked via bit 3 on I/O port 21h. On DOS 3.3+, IRQ3 is re-vectored to a task-switching routine, unless STACKS=0 is set in CONFIG.SYS. On MS-DOS and PC-DOS, the handler uses the IBM Interrupt Sharing Protocol when hooking IRQ3.

9.3.16 INT 0Bh – Missing Segment (#NP)

Type: Fault

Introduced: 80286

Triggers if the present flag of a segment or gate descriptor is cleared. Typically in a task switch.

9.3.17 INT 0Ch – Serial Communications via IRQ4 (COM1)

Type: Unknown

Triggered if COM1 requires attention, that is, if the UART has been programmed to generate interrupts. May be masked via bit 4 on I/O port 21h. On DOS 3.3+, IRQ4 is re-vectored to a task-switching routine, unless STACKS=0 is set in CONFIG.SYS. On MS-DOS and PC-DOS, the handler uses the IBM Interrupt Sharing Protocol when hooking IRQ4.

9.3.18 INT 0Ch – Stack Fault (#SS)

Type: Fault

Introduced: 80286

Triggered if a instruction related to the SS register failed, such as POP, PUSH, CALL, etc., loading the SS register when the stack segment is missing (marked not-present) in a task-switch, or the stack overflowed.

It can also be generated in REAL mode when accessing a WORD operand at SS:FFFFh.

9.3.19 INT 0Dh – Disk Driver Interface via IRQ5 (IBM PC XT)

Type: Unknown

Description for this article is unknown.

9.3.20 INT 0Dh – General Protection Exception (#GP)

Type: Fault

Introduced: 80286

Triggered when the interrupt detects a general protection violation which does not fit under other exceptions, like reading data from an execute-only segment.

9.3.21 INT 0Eh – Diskette Driver Interface via IRQ6

Type: Unknown

Triggered by the floppy disk controller when completing an operation. Default handler is located at F000h:EF57h in the IBM PC. May be masked off via bit 6 on I/O port 21h. On DOS 3.3+, IRQ6 is re-vectored to a task-switching routine, unless STACKS=0 is set in CONFIG.SYS. On MS-DOS and PC-DOS, the handler uses the IBM Interrupt Sharing Protocol when hooking IRQ6.

9.3.22 INT 0Eh – Page Fault (#PF)

Type: Fault

Introduced: 80386

Triggered in vm8086 and protected modes when attempting to access a 4K memory page where the table entry has the present bit cleared.

9.3.23 INT 0Fh – Printer Driver Interface via IRQ7

Type: Unknown

Triggered by the LPT1 printer adapter when the printer becomes in a ready state. Typically, the 8259 interrupt controller generates the interrupt, not the printer adapter.

9.4 BIOS Interrupts

BIOS interrupts are typically used to access hardware features, such as setting the cursor position, within an IBM PC.

Interrupts 1Dh to 1Fh (Video, Diskette, and Video Graphics parameters) are interrupt vectors that points to data instead of instructions.

9.4.1 INT 10h – Video Driver Interface

The video driver interface under the IBM PC takes an argument in AH register to select the appropriate function.

9.4.1.1 AH=00H – SET VIDEO MODE

Input:

AH=00h
AL=Video Mode

Return:

AL=Video Mode Flag (Pheonix and AMI BIOS, see DOS Video Modes table)

Sets video mode, where AL is typically a value from 00h to 07h. Default is 03h.

Resolution follows a WIDTHxHEIGHTxCOLOR[g] notation, where g indicates a monochrome color scheme.

Note: DD-DOS runs strictly in text mode, thus why only the supported modes are shown, and only screen sizes are taken into account when the mode is set.

AL	Screen Size	Character Size	Resolution	Pages	Address	Adapter
00h	40x25	8x8	320x200x16g	8	B800h	CGA
01h	40x25	8x8	320x200x16	8	B800h	CGA
02h	80x25	8x8	640x200x16g	4	B800h	CGA
03h	80x25	8x8	640x200x16	4	B800h	CGA
04h	40x25	8x8	320x200x4	--	B800h	CGA
05h	40x25	8x8	320x200x4g	--	B800h	CGA
06h	80x25	8x8	640x200x2g	--	B800h	CGA
07h	80x25	9x14	720x350x1	--	B000h	VGA

Table 36: Video Modes

The return value is a byte flag.

Bit	Description
1	Mouse port present Set if supported in DD-DOS
4	Programmable video clock, unsupported Cleared by default

Table 37: Video Mode Flag

9.4.1.2 AH=01H – SET TEXT-MODE CURSOR SHAPE

Unsupported.

9.4.1.3 AH=02H – SET CURSOR POSITION

Input:

AH=02h
BH=Page number
DH=Row, 0 being top
DL=Column, 0 being left

Set the current cursor position.

Note: DD-DOS currently does not support Pages.

9.4.1.4 AH=03H – GET CURSOR POSITION AND SIZE

Input:

BH=Page number

Return:

AX=0000h (Phoenix BIOS)
CH=Start scan line
CL=End scan line
DH=Row
DL=Column

Get cursor position and size by page number. DD-DOS does not support scan lines.

9.4.1.5 AH=04H – LIGHT PEN POSITION

Return:

AH=Trigger flag (0=lifted, 1=down)
DH=Row
DL=Column
CH=Pixel row (Where VIDEO MODE 04h-06h)
CX=Pixel row (Where pixel rows >200)
BX=Pixel column

Read the position of the light pen, if supported. This BIOS subroutine does not work under VGA adapters. On a CGA adapter, return columns are typically multiples of 2 (Video Mode 00h-01h, 04h-05h), or multiples of 4 (Video Mode 02h-03h, 06h).

Note: DD-DOS aims to support the light pen regardless of video modes.

9.4.1.6 AH=06H – SELECT DISPLAY PAGE

Input:

AL=Page number

Selects the page to display and set as active.

Note: DD-DOS does not support this.

9.4.2 INT 11h – Equipment Check

Return:

AX=Equipment list flags (see Equipement List Flags table)

Returns a 16-bit value in AX as flags.

Bit	Default value	Description
0	Cleared	Set if floppy drives(s) installed
1	Cleared	Set if 80x87 coprocessor is installed
3:2	?	Number of 16K (PC), or 64K (PC XT) memory banks installed
5:4	10b	Initial video mode 00b – EGA, VGA, or PGA 01b – 40x25, color 10b – 80x25, color 11b – 80x25, monochrome
7:6	0	Number of floppy drives installed, relates to bit 0
8	--	Unused
11:9	0	Number of serial ports installed (COM)
12	Cleared	Game port installed
13	Cleared	Internal modem installed
15:14	0	Number of parallel ports installed

Table 38: Equipment Check Flags

9.4.3 INT 12h – Get Memory Size

Return:

AX=Memory size in KB

Returns in AX the total memory size in kilobytes (MEMORYSIZE / 1024).

9.4.4 INT 13h – Disk Operations

9.4.4.1 AH=00H – RESET DISK SYSTEM

Input:

AH=00h
DL=Drive number

Return:

AH=Status (00h if successful)
CF clear if successful; Otherwise set

Forces the disk controller to re-calibrate drive heads (seek to track 0).

If bit 7 is set in DL, both hard disks and floppy disk drives are reset.

Note: DD-DOS will always return 0 with a cleared CF.

9.4.4.2 AH=01H – GET STATUS OF LAST OPERATION

Input:

AH=01h
DL=Drive number

Return:

AH=Status
CF clear if successful; Otherwise set

Get the status of the last disk operation.

If bit 7 in DL is set, the hard disk is checked; Otherwise, the floppy disk is checked.

Note: DD-DOS only supports a limited range of statuses.

AH	Status
00h	Successfully completed
01h	Invalid DL parameter
02h	Address mark not found
03h	Disk is write-protected
04h	Sector not found, or read error
05h	Reset failed
06h	Floppy disk changed
07h	Hard disk drive parameter activity failed
08h	DMA overrun
09h	Data boundary error (DMA across 64K or more than 80h sectors)
0Ah	Hard disk bad sector
0Bh	Hard disk bad track
0Ch	Unsupported track or invalid media
0Dh	Invalid number of sectors on format (PS/2)
0Eh	Hard disk control data address mark detected

AH	Status
0Fh	Hard disk DMA arbitration level out of range
10h	Unrecoverable CRC or ECC error on read
11h	Hard disk corrected ECC data
20h	Controller failure
31h	No media in drive (IBM and MS INT 0Dh extension)
40h	Seek failed
80h	Timeout, not ready
AAh	Drive not ready
B0h	Volume not locked in drive (INT 0Dh extension)
B1h	Volume locked in drive (INT 0Dh extension)
B2h	Volume not removable (INT 0Dh extension)
B3h	Volume in use (INT 0Dh extension)
B4h	Lock count exceeded (INT 0Dh extension)
B5h	Eject request failed (INT 0Dh extension)
B6h	Volume is read protected (INT 0Dh extension)
BBh	Hard disk undefined error
CCh	Hard disk write fault
E0h	Hard disk status register error
FFh	Hard disk operation failed

Table 39: Disk Last Operation Status Return Value

9.4.4.3 AH=02H – READ SECTORS INTO MEMORY

Input:

AH=02h
AL=Number of sectors to read
CH=Low eight bits of cylinder number
CL=Sector number
Bits 0-5 is sector number (1-63)
Bits 6-7 is the high bits of cylinder number (hard disk only)
DH=Head number
DL=Drive number
ES:BX → Data buffer

Return:

AH=Status
AL=Number of sectors read
CF cleared if successful; Otherwise set on error

Read sectors directly from disk into the buffer specified at ES:BX. If bit 7 of DL is set, a hard disk is selected. If return status is 11h (corrected ECC error), then AL is its burst length. For status codes, see table on §9.4.4.2.

Note: DD-DOS does not support this function.

9.4.4.4 AH=03H – WRITE DISK SECTORS

Input:

AH=03h
AL=Number of sectors to write
CH=Low eight bits of cylinder number
CL=Sector number
Bits 0-5 is sector number (1-63)
Bits 6-7 is the high bits of cylinder number (hard disk only)
DH=Head number
DL=Drive number
ES:BX → Data buffer

Return:

AH=Status
AL=Number of sectors written
CF cleared if successful; Otherwise set on error

Write sectors directly to disk from the buffer specified at ES:BX. If bit 7 of DL is set, a hard disk is selected. If return status is 11h (corrected ECC error), then AL is its burst length. For status codes, see table on §9.4.4.2.

9.4.4.5 AH=04H – VERIFY DISK SECTORS

Input:

AH=04h
AL=Number of sectors to verify
CH=Low eight bits of cylinder number
CL=Sector number
Bits 0-5 is sector number (1-63)
Bits 6-7 is the high bits of cylinder number (hard disk only)
DH=Head number
DL=Drive number
ES:BX → Data buffer

Return:

AH=Status
AL=Number of sectors verified
CF cleared if successful; Otherwise set on error

Verify sectors directly from disk to the buffer specified at ES:BX. If bit 7 of DL is set, a hard disk is selected. For status codes, see table on §9.4.4.2.

9.4.4.6 AH=05H – FLOPPY, FORMAT TRACK

Input:

AH=05h
AL=Number of sectors to format
CH=Track number
DH=Head number
DL=Drive number
ES:BX → Buffer location

Return:

AH=Status
CF cleared if successful; Otherwise set on error

Format a track on a floppy disk drive. For status codes, see table on §9.4.4.2

9.4.4.7 AH=05H – FLOPPY, FORMAT TRACK

Input:

AH=05h
AL=Number of sectors to format
CH=Track number
DH=Head number
DL=Drive number (bit 7 must be set)
ES:BX → Buffer location

Return:

AH=Status
CF cleared if successful; Otherwise set on error

Format a track on a floppy disk drive. For status codes, see table on §9.4.4.2.

9.4.5 INT 14h – Serial Port Interface

9.4.5.1 AH=00H – INITIALIZE PORT

Input:

AH=00h
AL=Port parameters (see Serial Port Initialization Parameters)
DX=Port number (00-03h)

Return:

AH=Line status (see Serial Port Status)
AL=Modem status (see Modem Status Flags)

Initializes a specified serial port with parameters.

Bit	Default	Description
1:0	11b	Data bits 00b – 5 bits 01b – 6 bits 10b – 7 bits 11b – 8 bits

Bit	Default	Description
2	Set	Stop bits is active if the bit is set; Otherwise inactive
4:3	00b	Parity 00b, 10b – None 01b – Odd 11b – Even
7:5	111b	Data rate 000b – 110 bps 001b – 150 bps 010b – 300 bps 011b – 600 bps 100b – 1200 bps 101b – 2400 bps 110b – 4800 bps 111b – 9600 bps

Table 40: Serial Port Initialization Parameters

AH	Description
00h	Port available
01h	Port exists, but already in use
02h	Port doesn't exist

Table 41: Serial Port Statuses

Modem reports are not supported under DD-DOS, therefore AL will always return 00h.

Bit	Default	Description
0	Clear	Delta clear to send
1	Clear	Delta data set ready
2	Clear	Trailing edge of ring indicator
3	Clear	Delta carrier detect
4	Clear	Clear to send
5	Clear	Data set ready
6	Clear	Ring indicator
7	Clear	Carrier detect

Table 42: Modem Status Flags

9.4.5.2 AH=01H – WRITE CHARACTER

Input:

AH=01h

AL=Character
DX=Port number

Return:

AH=Status (see Serial Port Statuses table)

Write a single ASCII character to the specified serial port. Bit 7 of AH is cleared if successful, otherwise set on error.

9.4.5.3 AH=02H – READ CHARACTER

Input:

AH=02h
DX=Port number

Return:

AH=Status (see Serial Port Statuses table)
AL=Character

Read a single ASCII character from the specified serial port. Bit 7 of AH is cleared if successful, otherwise set on error.

9.4.5.4 AH=03H – GET STATUS

Input:

AH=03h
DX=Port number

Return:

AH=Port status (see Serial Port Statuses)
AL=Modem status (see Modem Status Flags)

Get the status of a serial port.

9.4.6 INT 16h – Keyboard Interface

9.4.6.1 AH=00H – GET KEYSTROKE

Input:

AH=00h

Return:

AH=BIOS scan code
AL=ASCII character

Gets a keystroke.

Note: DD-DOS emulates PS/2 scan codes from a IBM PC keyboard as much as possible.

9.4.6.2 AH=01H – CHECK KEYSTROKE

Input:

AH=01h

Return:

AH=BIOS scan code

AL=ASCII character
ZF cleared if available; Otherwise set if no keystrokes are available

Check if a keystroke is present from the keyboard buffer.

Note: DD-DOS may not support this.

9.4.6.3 AH=02H – GET MODIFIER FLAGS

Input:

AH=02h

Return:

AL=Flags (see Modifier Keys Flags)

Returns the flags of several modifier keys. DD-DOS does not destroy AH.

Note: DD-DOS does not support the Insert key toggle.

Bit	Description
0	Right Shift key pressed
1	Left Shift key pressed
2	Ctrl key pressed (either)
3	Alt key pressed (either)
4	ScrollLock is active
5	NumLock is active
6	CapsLock is active
7	Insert is active

Table 43: Modifier Keys Flags

9.4.6.4 AH=09H – GET KEYBOARD FEATURES

Input:

AH=09h

Return:

AL=Keyboard functions supported (see Keyboard Features Flags)

Get keyboard functionality. This function is only available if bit 6 of the second feature byte return INT 15h AH=C0h is set. DD-DOS does not destroy AH.

Bit	Description
0	INT 16h AX=0300h supported
1	INT 16h AX=0304h supported
2	INT 16h AX=0305h supported
3	INT 16h AX=0306h supported
4	INT 16h AX=0A00h supported
5	INT 16h AH=10h-12h supported (enhanced keyboard support)
6	INT 16h AH=20h-22h supported (122-key keyboard support)
7	Reserved, cleared

Table 44: Keyboard Features Flags

9.4.6.5 AH=A0H – GET KEYBOARD ID

Input:

AH=A0h

Return:

BX=Keyboard ID (see Keyboard IDs)

Get the keyboard ID.

Note: DD-DOS only returns 86ABh.

Value	Keyboard description
0000h	No keyboard attached
41ABh	Japanese "G" keyboard (translate mode) or MF2 Keyboard (usually in translate mode)
54ABh	54ABh Japanese "P" keyboard (translate mode)
83ABh	Japanese "G" keyboard (pass-through mode) or MF2 Keyboard (pass-through mode)
84ABh	Japanese "P" keyboard (pass-through mode)
86ABh	122-key keyboard
90ABh	old Japanese "G" keyboard
91ABh	old Japanese "P" keyboard
92ABh	old Japanese "A" keyboard

Table 45: Keyboard IDs

9.4.7 INT 19h – Bootstrap Loader

Reboots the system without cleaning memory (including interrupt vectors). This interrupt causes a system hang if any Terminate and Stay Resident programs are loaded in memory.

9.4.8 INT 1Ah – System Time Functions

9.4.8.1 AH=00H – GET SYSTEM TIME

Input:

AH=00h

Return:

CS:DX = Number of clock ticks since midnight
AL=Midnight flag, set if midnight passed since last time read

Get the number of click ticks (~18.2 ticks/second).

Note: DD-DOS follows the MS-DOS 5.0 behavior, where AL is assumed to be a day rollover counter than a flag.

9.4.8.2 AH=01H – SET SYSTEM TIME

Input:

AH=01h
CX:DX=Number of clock ticks since midnight

Sets the system time to clock ticks in CX:DX.

Note: DD-DOS does not support this function.

9.4.8.3 AH=02H – GET REAL-TIME CLOCK TIME

Input:

AH=02h

Return:

CH=Hour (BCD)
CL=Minutes (BCD)
DH=Seconds (BCD)
DL=Daylight savings flag (clear=standard time, set=daylight time)
CF set on error, cleared otherwise

Get the time from the real-time clock.

9.4.8.4 AH=03H – SET REAL-TIME CLOCK TIME

Input:

AH=03h
CH=Hour (BCD)
CL=Minutes (BCD)
DH=Seconds (BCD)
DL=Daylight savings flag (clear=standard time, set=daylight time)

Set the time from the real-time clock.

Note: DD-DOS does not support this function.

9.4.8.5 AH=04H – GET REAL-TIME CLOCK DATE

Input:

AH=04h

Return:

CH=Century (BCD)
CL=Year (BCD)
DH=Month (BCD)
DL=Day (BCD)
CF set on error, cleared otherwise

Get the date from the real-time clock.

Note: DD-DOS is Y2K compliant.

9.4.8.6 AH=05H – SET REAL-TIME CLOCK DATE

Input:

AH=05h
CH=Century (BCD)
CL=Year (BCD)
DH=Month (BCD)
DL=Day (BCD)

Set the date from the real-time clock.

Note: DD-DOS does not support this function.

9.4.8.7 AH=06H – SET ALARM

Input:

AH=06h
CH=Hour (BCD)
CL=Minutes (BCD)
DH=Seconds (BCD)

Return:

CF set on error (alarm already set or clock stopped for update), cleared otherwise

Sets an alarm for every laps of time, occurring every 24 hours until turned off. For example, an alarm that goes off every minute, once a minute, you could use CH=FFh, CL=FF, and DH=00h.

The clock handler is invoked via INT 4Ah.

9.4.8.8 AH=07H – CANCEL ALARM

Input:

AH=07h

Cancel the set alarm (via AH=06h). This does not disable the real-time clock's IRQ.

9.4.9 INT 1Bh – CTRL+BREAK Handler

This interrupt is automatically called when INT 09h determines that CTRL+BREAK has been pressed.

Note: DD-DOS sets CTRL+C flag and invokes INT 23h the next time DD-DOS checks for CTRL+C.

9.4.10 INT 1Ch – System Timer Tick

This interrupt is automatically called on each clock tick by the INT 08 handler. This is the preferred interrupt to chain when a program needs to be invoked regularly.

9.4.11 INT 1Dh – Video Parameters

This interrupt is only a vector that points to a buffer at F000:F0A4h with a Video Parameters Structure.

Offset	Size	Description
0	16	6845 register values for modes 00h and 01h
16	16	6845 register values for modes 02h and 03h
32	16	6845 register values for modes 04h and 05h
48	16	6845 register values for modes 06h and 07h
64	2	Bytes in video buffer for modes 00h and 01h (0800h)
66	2	Bytes in video buffer for modes 02h and 03h (1000h)
68	2	Bytes in video buffer for modes 04h and 05h (4000h)
70	2	Bytes in video buffer for modes 06h (4000h)
72	8	Columns on screen for each of modes 00h through 07h
80	8	CRT controller mode bytes for each of modes 00h through 07h

Table 46: Video Parameters Structure

9.4.12 INT 1Eh – Diskette Parameters

This interrupt is only a vector that points to a 16-byte buffer at F000:EFC7h with a Diskette Parameter Structure.

Offset	Size	Description
0	1	Command 1
1	1	Command 2
2	1	Motor off time
3	1	Bytes per sector

Offset	Size	Description
4	1	Sectors per track
5	1	Gap length between sectors
6	1	Data length (always FFh)
7	1	Gap length for FORMAT
8	1	Fill byte for FORMAT
9	1	Head settle time
10	1	Motor-on start time
11	1	Max track number for selected drive
12	1	Data transfer rate
13	1	Unused
14	1	Unused

Table 47: Diskette Parameter Structure

9.4.13 INT 1Fh – Video Graphics Characters

This interrupt is only a vector that points to a buffer at F000:FA6Eh.

The buffer is 1024 bytes of graphical data, 8 bytes for each character, 80h to FFh.

Note: DD-DOS does not support custom graphical characters for foreign characters.

9.4.14 INT 41h – Fixed Disk Parameter Table

This interrupt is only a vector that points to a buffer at F000:E401h with a Fixed Disk Parameter Structure.

Note: DD-DOS does not support this and will use default values.

Offset	Size	Description
00h	2	Number of cylinders
02h	1	Number of heads
03h	2	Starting reduced write current cylinder
05h	2	Starting write pre-compensation cylinder number
07h	1	Maximum ECC burst length
08h	1	Control byte (DTE_CONTROL), see XT Fixed Disk Control Byte table
09h	1	Standard timeout
0Ah	1	Formatting timeout (XT and WD1002)
0Bh	1	Timeout for checking drive (XT and WD1002)
0Ch	2	Cylinder landing zone
0Eh	1	Number of sectors per track
0Fh	1	Reserved

Table 48: Fixed Disk Parameter Structure

Bit(s)	Description
2:0	Drive step speed b000 – 3 ms b100 – 200 ms b101 – 70 ms (default) b110 – 3 ms b111 – 3 ms
5:3	Unused
6	Disable ECC retries
7	Disable access retries

Table 49: XT Fixed Disk Control Byte Fields

9.5 DOS Interrupts

9.5.1 INT 20h – Terminate Program

Input:

CS=PSP Segment

This function terminates a running program and sets ERRORLEVEL to 0. For later MS-DOS versions (2.0 and higher), Microsoft recommends using INT 21h/AH=4Ch.

9.5.2 INT 21h – MS-DOS Services

This interrupt and all their subroutines are explained in MS-DOS Services.

9.5.3 INT 22h – Program Termination Address

This vector does not point to an interrupt handler; it specifies the address of the next routine after a terminating program.

The vector is restored from DWORD at PSP+0Ah during termination, and then a far jump is performed to address at INT 22h. Usually points to the instruction following INT 21h/AH=4Bh which loaded the program.

9.5.4 INT 23h – CTRL+C/CTRL+BREAK Handler

This interrupt is invoked whenever DOS detects CTRL+C or CTRL+BREAK keypresses; it should never be called directly. ERRORLEVEL is set to 0, all registers are preserved, and the interrupted DOS call is restarted.

9.5.5 INT 24h – Critical Error Handler

Return:

AL=Action code (see Action Codes)

Invoked when a critical, usually hardware, error occurs; should never be called directly. Please note that SS, SP, DS, ES, BX, CX, and DX are preserved.

The error handler is called with the following parameters:

AH=Type and processing flags (see Error Types)
AL=Drive number if AH[7] is cleared
BP:SI → Device driver header (see Device Driver Header), BP:SI+4[15] is set if it's a character device
DI=If AH[7] bit is set, DI contains the error code

Stack contains the following:

DWORD	Return address for INT 24h
WORD	Flags pushed by INT 24h
WORD	Original AX on entry to INT 21h
WORD	BX
WORD	CX
WORD	DX
WORD	SI
WORD	DI
WORD	BP
WORD	SP
WORD	DS
WORD	ES

DWORD Return address for INT 21h
WORD Flags pushed by INT 21h

Bit	Description
0	Set on write error, cleared if on read error
2:1	Disk area of error b00 – DOS area b01 – FAT b10 – Root directory b11 – Data area
3	Failed allowed
4	Retry allowed
5	Ignore allowed
6	Unused
7	Type. If set, it's a disk I/O error, otherwise if cleared, bad FAT image in memory

Table 50: INT 24h Error Type And Processing Flags

AH	Description
0	Ignore error and continue processing request
1	Retry operation
2	Terminate program, even if it was called from INT 21h/AH=4Ch
3	Fail system call in progress

Table 51: INT 24h Action Code

Offset	Size	Description
0	4	Pointer to next driver, this field is set to FFFFh if it's the last driver
4	2	Device attributes (see Character Device Attributes or Block Device Attributes)
6	2	Device strategy entry point (called with ES:BX → request header)
8	2	Device interrupt entry point

Offset	Size	Description
10	8	(character device) Space-padded device name, these may include: "$PCMATA" PCMCIA driver (PCMATA.SYS) "AHADDVR" Adaptec SCSI disk driver (ASPIDISK.SYS) "DBLSPAC" MS DoubleSpace or DriveSpace "DSKREET" NortonUtils v5 (or later) Diskreet "GFS " LapLink III device driver (DD.BIN) "SIDExxx" PCMCIA driver (ATADRV.EXE) "STAC-CD" Stacker/Stacker Anywhere (block device) 1-byte number of sub-units (drives) supported by driver, the rest is usually unused or a signature
18	2	Reversed, must be 0000h
20	1	(CD-ROM) Drive letter (A: being 01h, etc.), must be initially 00h
21	1	(CD-ROM) Number of units
22	6	(CD-ROM) Signature, "MSCDnn" where nn is version, usually 00h

Table 52: DOS Device Driver Header Structure

Bit(s)	Description
0	Device is standard input
1	Device is standard output
2	Device if NUL
3	Device is CLOCK$
4	Device is special
5	Reserved
6	(DOS 3.2+) Generic IOCTL call supported (driver command 13h)
7	(DOS 5.0+) Generic IOCTL check call supported (driver command 19h)
10:8	Reserved
11	(DOS 3.0+) OPEN/CLOSE/RemMedia calls supported
12	Reserved
13	(DOS 3.0+) Output until busy feature supported
14	IOCTL supported
15	Always set; indicating a character device

Table 53: Character Device Attributes

Bit(s)	Description
0	Reserved
1	(DOS 3.31+) If set, supports 32-bit sector addressing
5:2	Reserved
6	(DOS 3.2+) Generic IOCTL call supported (driver command 13h)
7	(DOS 5.0+) Generic IOCTL check call supported (driver command 19h)
8	(DOS 3.3) Set for DRIVER.SYS for new drives[1]
9	(DOS 3.3) DRIVER.SYS for new drives – Set if direct I/O not allowed[2]
10	Reserved
11	(DOS 3.0+) OPEN/CLOSE/RemMedia calls supported
12	Network device, and the device is remote
13	Non-IBM format
14	IOCTL supported
15	Always cleared; indicating a block device

Table 54: Block Device Attributes

9.5.6 INT 25h – Absolute Disk Read (< 32M)

Input:

AL=Drive number (A: being 00h, etc.)
CX=Number of sectors to read (must not be FFFFh)
DX=Starting logical sector number (0000h being highest sector)
DS:BX → Data buffer

Return:

AH=Status (see Absolute Read And Write Error Codes)
AL=Error code (see MS-DOS Critical Error Codes)
CF set on error, cleared if successful

Directly read an amount of data, in sectors, to a specified buffer in memory. This call bypasses the DOS filesystem, and cannot be used to read data where partitions are over 32M.

If AX is set to 0207h, there are more than 64K sectors on drive, and the user must use the newer style call, with AL[7] set.

Note: DD-DOS does not support this function.

1 Uncertain
2 Uncertain

Code	Description
01h	Bad command
02h	Bad address mark
03h	(INT 26h) Write-protected disk
04h	Requested sector not found
08h	DMA failure
10h	Data error, bad CRC
20h	Controller failed
40h	Seek operation failed
80h	Device failed to respond, timed out

Table 55: Absolute Read And Write Error Codes

Code	Description
00h	Write-protection violation
01h	Unknown unit for driver
02h	Drive not ready
03h	Unknown command given to driver
04h	Data error, bad CRC
05h	Bad device driver request structure length
06h	Seek error
07h	Unknown media type (non-DOS disk)
08h	Sector not found
09h	Printer out of paper
0Ah	Write fault
0Bh	Read fault
0Ch	General
0Dh	(DOS 3.0+) Sharing violation
0Eh	(DOS 3.0+) Lock violation
0Fh	Invalid disk change or wrong disk
10h	Uncertain media (DOS 3.0+) FCB unavailable

Code	Description
11h	Character call interrupted (DOS 3.0+) Sharing buffer overflow
12h	(DOS 4.0+) Code page mismatch
13h	(DOS 4.0+) Out of input
14h	(DOS 4.0+) Insufficient disk space

Table 56: MS-DOS Critical Error Codes

9.5.7 INT 25h/CX=FFFFh – Absolute Disk Read (32M-2047M)

Input:

CX=FFFFh
AL=Drive number
DS:BX → Disk read packet buffer (see MS-DOS Disk Packet Structure)

Return:

AH=Status (see Absolute Read And Write Error Codes)
AL=Error code (see MS-DOS Critical Error Codes)
CF set on error, cleared if successful

Directly read data from disk. This function bypasses the DOS filesystem.

For FAT32 drives, which may be up to 2T in size, use INT 21h/AX=7305h.

Note: DD-DOS does not support this function.

Offset	Size	Description
0	4	Sector number
4	2	Number of sectors to read
6	4	Transfer address

Table 57: MS-DOS Disk Packet Structure

9.5.8 INT 26h – Absolute Disk Write (< 32M)

Input:

AL=Drive number (A: being 00h, etc.)
CX=Number of sectors to read (must not be FFFFh)
DX=Starting logical sector number (0000h being highest sector)
DS:BX → Data buffer

Return:

AH=Status (see Absolute Read And Write Error Codes)
AL=Error code (see MS-DOS Critical Error Codes)
CF set on error, cleared if successful

Directly writes an amount of data, in sectors, to a disk. This call bypasses the DOS filesystem, and cannot be used to write data where partitions are over 32M.

If AX is set to 0207h, there are more than 64K sectors on drive, and the user must use the newer style call, with AL[7] set.

Note: DD-DOS does not support this function.

9.5.9 INT 26h/CX=FFFFh – Absolute Disk Write (32M-2047M)

Input:

CX=FFFFh
AL=Drive number
DS:BX → Disk write packet buffer (see MS-DOS Disk Packet Structure)

Return:

AH=Status (see Absolute Read And Write Error Codes)
AL=Error code (see MS-DOS Critical Error Codes)
CF set on error, cleared if successful

Directly read data from disk. This function bypasses the DOS filesystem.

For FAT32 drives, which may be up to 2T in size, use INT 21h/AX=7305h.

Note: DD-DOS does not support this function.

9.5.10 INT 27h – Terminate And Stay Resident

Input:

DX=Number of bytes to keep resident (max: FFF0h)
CS=PSP segment

Terminate a program but leave its code segment alive in memory. This function does not close open file handles.

This function is obsolete, Microsoft strongly recommends using INT 21h/AH=31h.

9.5.11 INT 28h – DOS Idle Interrupt

Input:

SS:SP=Top of MS-DOS stack for I/O functions

This interrupt is invoked every time the DOS character input function loops while waiting for input. The INT 28h handler may invoke any INT 21h function (MS-DOS Service), except functions from 00h to 0Ch.

Note: Under DD-DOS, this function is blocking, and may not be implemented.

9.5.12 INT 29h – Fast Console Output

Input:

AL=Character to display

Display one character at the current cursor position.

9.5.13 INT 2Eh – Pass Command To Command Interpreter

Input:

DS:SI → Command line packet to execute (see Command Line Packet)

Return:

CF cleared on success, otherwise set on error

This function passes a command line string to the current shell (usually COMMAND.COM) for execution, without having to load an additional copy of the current shell.

Note: DD-DOS does not follow the 4DOS standard of setting AX as error code. Setting CF is a custom DD-DOS standard.

Offset	Size	Description
0	1	Length of command string, excluding trailing CR character (Dh)
1	Variable	Command string, must finish with CR character

Table 58: DOS Command Line Packet

9.5.14 INT 2Fh – Print Functions

Please note that printing is currently not implemented in DD-DOS.

9.6 MS-DOS Services (INT 21h)

The MS-DOS Services, introduced in MS-DOS 1.0, takes the AH register as a parameter, selecting the requested service. The services are accessible via interrupt 21h. Microsoft recommends using the MS-DOS services whenever possible.

9.6.1 AH=00h – Terminate Program

Input:

AH=00h
CS=PSP segment

Terminates the running program. This function sets ERRORLEVEL to 0. Program flow continues at the address stored in INT 22h after the DOS cleanup phase.

9.6.2 AH=01h – Read Character From Standard Input, With Echo

Input:

AH=01h

Return:

AL=ASCII character

Read a key from the standard input stream (stdin), including the ENTER key (CR), and prints the character on screen. This function is blocking, but allows CTRL+C and CTRL+BREAK, where INT 23h is executed, if detected.

9.6.3 AH=02h – Write Character To Standard Output

Input:

AH=02h
DL=ASCII character to write

Return:

AL=Last character printed

Print a character to the standard output stream (stdout), on screen.

9.6.4 AH=03h – Read Character From Standard Auxiliary

Input:

AH=03h

Return:

AL=ASCII character

Read a character from the standard auxiliary stream (stdaux), including the ENTER key (CR). This function is blocking, but allows CTRL+C and CTRL+BREAK, where INT 23h is executed, if detected.

Standard Auxiliary is usually the first serial port (COM1).

9.6.5 AH=04h – Write Character To Standard Auxiliary

Input:

AH=04h
DL=ASCII character to write

Write a character to the standard auxiliary stream (stdaux). Allows CTRL+C and CTRL+BREAK, where INT 23h is executed, if detected.

Standard Auxiliary is usually the first serial port (COM1). If stdaux is busy, this function will wait.

9.6.6 AH=05h – Write Character To Standard Printer

Input:

AH=05h
DL=ASCII character to write

Write a character to the standard printer stream (stdprn). This function checks for CTRL+C and CTRL+BREAK, where INT 23h is then executed. If the printer is busy, this function will wait.

Standard Printer is usually the first serial port, and is usually redirected after DOS 2.0.

9.6.7 AH=06h – Direct Console Output

Input:

AH=06h
DL=ASCII character (except FFh, see Direct Console Input)

Return:

AL=Character output

Write a character to the standard output stream (stdout). This function does not check for CTRL+C and CTRL+BREAK.

9.6.8 AH=06h/DL=FFh – Direct Console Input, With Echo

Input:

AH=06h
DL=FFh

Return:

AL=00h, or ASCII character
ZF set on no available characters, cleared if a character is available

Read a character from the standard input stream (stdin) and echoes the character on screen, if not 00h. This function is blocking and does not check for CTRL+C and CTRL+BREAK.

If AL is set to 00h, an extended keycode had been pressed, which will be the result of the next call.

9.6.9 AH=07h – Direct Console Input, Without Echo

Input:

AH=07h

Return:

AL=ASCII character

Read a character from the standard input stream (stdin). This function is blocking and does not check for CTRL+C and CTRL+BREAK.

9.6.10 AH=08h – Character Input, Without Echo

Input:

AH=08h

Return:

AL=ASCII character

Read a key from the standard input stream (stdin), including the ENTER key (CR). This function is blocking, but allows CTRL+C and CTRL+BREAK, where INT 23h is executed, if detected.

9.6.11 AH=09h – Write String To Standard Output

Input:

AH=09h
DS:DX → Dollar-sign terminated string

Return:

AL=24h ('$')

Writes a dollar-sign terminated string to the standard output stream. This function checks for CTRL+C and CTRL+BREAK, and INT 23h is called if either are detected.

Note: DD-DOS limits strings to 255 characters.

9.6.12 AH=0Ah – Buffered Input

Input:

AH=0Ah
DS:DX → Input buffer

Read characters to a buffer. This function checks for CTRL+C and CTRL+BREAK, where INT 23h will be called, if detected.

Offset	Size	Description
0	1	Size of buffer (number of characters the buffer can hold). If set to 0, the function returns immediately without reading any input
1	1	On call, number of characters from last input On return, number of characters actually read, including CR
2	Variable	Character buffer

Table 59: DOS Input Buffer

9.6.13 AH=0Bh – Get Standard Input Status

Input:

AH=0Bh

Return:

AL=Status (00h if no characters available, FFh otherwise)

Gets the status of stdin. This function checks for CTRL+C and CTRL+BREAK, where INT 23h will be called, if detected.

9.6.14 AH=0Ch – Flush Buffer And Read Standard Input

Input:

AH=0Ch
AL=Standard input function to execute after flushing (AH for MS-DOS service)

Flush the stdin buffer and starts reading for input. If AL is not 01h, 06h, 07h, 08h, or 0Ah, the buffer is flushed and no input is attempted.

9.6.15 AH=0Dh – Disk Reset (Flush Disk Buffers)

Input:

AH=0Dh

Flush all disk buffers, forcing to write buffer data to disk. This function does not update the directory information.

9.6.16 AH=0Eh – Select Default Drive

Input:

AH=0Eh
DL=Drive (00h being A:, etc.)

Return:

AL=Number of potentially valid drive numbers

Selects the default drive for disk operations.

9.6.17 AH=0Fh – Open File Using FCB

Input:

AH=0Fh
DS:DX → Unopened FCB (see File Control Block Structure)

Return:

AL=Status (00h on success, FFh if file not found or access denied)

Open a file using a FCB in memory. The FCB contains the drive, filename, and extension fields needed to open a file. The other fields in the FCB are cleared.

Microsoft recommends AH=3Dh to open an existing file.

9.6.18 AH=10h – Close File Using FCB

Input:

AH=10h
DS:DX → Opened FCB

Return:

AL=Status (00h on success, FFh on failure)

Close a file handle using a FCB in memory. This function flushes disk buffers before closing.

9.6.19 AH=11h – Find First Matching File Using FCB

Input:

AH=11h
DS:DX → Unopened FCB (see File Control Block Structure)

Return:

AL=Status (00h on success, FFh on no matching filename or bad FCB)

Finds the first match given a FCB. The drive, filename, and extension fields must be set while the rest of the FCB must be cleared. Upon finding a file, a DOS directory entry is created after the driver letter, which is similar to the FCB.

This function accepts the asterisk ('*') wildcard for filenames and extensions.

Offset	Size	Description
0	8	Space-padded filename First character is set to E5h for deleted files
8	3	Space-padded file extension
11	1	Attributes
12	10	Reserved
22	2	Creation or modified time stamp
24	2	Creation or modified date stamp
26	2	Starting Cluster number
28	4	File size

Table 60: DOS Directory Entry Structure

9.6.20 AH=12h – Find Next Matching File Using FCB

Input:

AH=12h
DS:DX → Unopened FCB

Return:

AL=Status (00h on success, FFh on no matching filename or bad FCB)

This function assumes INT 21h/AH=11h has been called prior, and is used in finding a matching file.

9.6.21 AH=13h – Delete File Using FCB

Input:

AH=13h
DS:DX → Unopened FCB

Return:

AL=Status (00h on success, FFh when no matching file, locked, or were all read-only)

Delete a file using an unopened FCB. Wildcards are allowed with '?'.

9.6.22 AH=14h – Sequential Read From FCB

Input:

AH=14h
DS:DX → Opened FCB

Return:

AL=Status (see INT 21h/AH=14h Statuses)

Read a record with the size specified in the FCB, beginning at the current file position, then updates the current block and the current record in the FCB. If a record was read partially, the rest of the record is zero-padded to the full size.

AL	Description
0	Successful
1	End of file (EOF), no more data
2	DTA is too small
3	End of file (EOF), partially read record

Table 61: INT 21h/AH=14h Statuses

9.6.23 AH=15h – Sequential Write From FCB

Input:

AH=15h
DS:DX → Opened FCB

Return:

AL=Status (see INT 21h/AH=15h Statuses)

Write a record of the specified size in the FCB at the current file position.

AL	Description
0	Successful
1	Disk full or read-only
2	DTA is too small

Table 62: INT 21h/AH=15h Statuses

9.6.24 AH=16h – Create Or Truncate File Using FCB

Input:

AH=16h
DS:DX → Unopened FCB

Return:

AL=Status (00h for success, FFh being directory full or file exists and is read-only or locked)

Create a file using an unopened FCB. If the file exists, it is truncated to zero length (empty file). Wildcards are not supported.

This function is also used to label a disk, at the disk's root directory.

9.6.25 AH=17h – Rename File Using FCB

Input:

AH=17h
DS:DX → Modified FCB (see Modified FCB Structure)

Return:

AL=Status (00h on success, FFh for no matching file, file is read-only, or new name exists)

Rename a existing file. '?' and '*' wildcards are allowed.

The rest of the FCB follows the standard FCB structure.

Offset	Size	Description
0	1	Drive designator
1	8	Original file name, space-padded
9	3	Original file extension, space-padded
17	8	New file name, space-padded
25	3	New extension, space-padded

Table 63: Modified FCB Structure

9.6.26 AH=19h – Get Current Default Drive

Input:

AH=19h

Return:

AL=Drive number (0 for A:, etc.)

Get the current default drive.

9.6.27 AH=1Ah – Set Disk Transfer Area Address

Input:

AH=1Ah
DS:DX → DTA

Set the address of the Disk Transfer Area for DOS. The address is usually PSP:80h when a program starts.

9.6.28 AH=1Bh – Get Allocation Table Information

Input:

AH=1Bh

Return:

AL=Sectors per cluster
CX=Bytes per sector
DX=Clusters on disk
DS:BX → Media Descriptor Byte (FAT)

Get information of the allocation table for the default disk.

9.6.29 AH=1Ch – Get Allocation Table Information For Specified Drive

Input:

AH=1Ch
DL=Drive number (0 for A:, etc.)

Return:

AL=Sectors per cluster
CX=Bytes per sector
DX=Clusters on disk
DS:BX → Media Descriptor Byte (FAT)

Get information of the allocation table for a specified drive.

9.6.30 AH=1Fh – Get Location To Current Drive Parameter Table

Input:

AH=1Fh
DL=Drive numner (0 for default, 1 for A:, etc.)

Return:

AL=Status (00h on success, FFh where drive does not exist)
DS:BX → Drive Parameter Table

Get the address location of the drive parameter table.

9.6.31 AH=21h – Random Read Using FCB

Input:

AH=21h
DS:DX → Opened FCB

Return:

AL=Status (see INT 21h/AH=14h Statuses)

Read a random record from the opened file. This function does not update the position field.

9.6.32 AH=22h – Random Write Using FCB

Input:

AH=22h
DS:DX → Opened FCB

Return:

AL=Status (see INT 21h/AH=15h Statuses)

Write at a random record specified data. This function does not update the position field.

9.6.33 AH=23h – Get File Size Using FCB

Input:

AH=23h
DS:DX → Unopened FCB

Return:

AL=Status (00h on success, FFh if not found)

Get the file using an unopened FCB. This function determines the number of records in a file, and updates the FCB.

9.6.34 AH=24h – Set Relative Record Field For FCB

Input:

AH=24h
DS:DX → Opened FCB

This function computes the random record depending on the current sequential block and record fields. Usually used when switching from sequential to random access.

9.6.35 AH=25h – Set Interrupt Vector

Input:

AH=25h
AL=Interrupt number
DS:DX → Interrupt handler

Safely change interrupt vectors.

Note: DD-DOS handles interrupts statically. This function may not work.

9.6.36 AH=26h – Create PSP

Input:

AH=26h
DX=Segment address for new PSP

Create a new Program Segment Prefix. DD-DOS takes the CS segment of the PSP to copy. The parent process is responsible for allocating memory necessary for the child process.

9.6.37 AH=27h – Random Block Read Using FCB

Input:

AH=27h
CX=Number of records to read
DS:DX → Opened FCB

Return:

AL=Status (see INT 21h/AH=14h Statuses)
CX=Number of records read (when AL is 00h or 03h)

Read a block of data at a random record into the DTA.

9.6.38 AH=28h – Random Block Write Using FCB

Input:

AH=27h
CX=Number of records to write
DS:DX → Opened FCB

Return:

AL=Status (see INT 21h/AH=15h Statuses)
CX=Number of records written

Write a block of data at a random record to the file.

9.6.39 AH=29h – Parse Filename Using FCB

Input:

AH=29h
AL=Control parsing flags (see INT 21h/AH=29h Control Parsing Flags)
DS:SI → Filename string
ES:DI → Unopened FCB

Return:

AL=Status (00h on success (01h with wildcards), FFh on failure (drive number))

Parse a string in memory to an unopened FCB in memory. The filename may have '?' and '*' wildcards and must not be a path.

Note: DD-DOS limits to 8 characters for the filename and 3 characters for the extension, including the period, a limit of 12 characters in total.

Bit	Description
0	If set, ignore leading separators If cleared, do not ignore leading separators
1	If set, modify drive number in FCB if specified, instead of 0 If cleared, modify drive number regardless
2	If set, modify filename in FCB if specified, instead of spaces If cleared, modify filename regardless
3	If set, modify extension in FCB if specified, instead of spaces If cleared, modify extension regardless
7:4	Reserved

Table 64: INT 21h/AH=29h Control Parsing Flags

9.6.40 AH=2Ah – Get System Date

Input:

AH=2Ah

Return:

CX=Year
DH=Month
DL=Day
AL=Day of the week (where 00h is Sunday)

Get the system date.

9.6.41 AH=2Bh – Set System Date

Input:

AH=2Bh
CX=Year
DH=Month (1-12)
DL=Day (1-31)

Return:

AL=Status (00h on success, FFh on invalid date)

Set the system date. This function also sets the virtual CMOS clock, if present.

Note: DD-DOS does not support this function.

9.6.42 AH=2Ch – Get System Time

Input:

AH=2Ch

Return:

CH=Hour
CL=Minute
DH=Second
DL=hundredth seconds (1/100)

Get the system time.

(Windows) DL is returned with wMilliseconds.

(Linux) DL is returned with tv_usec.

9.6.43 AH=2Dh – Set System Time

Input:

AH=2Dh
CH=Hour (0-23)
CL=Minute (0-59)
DH=Second (0-59)
DL=Hundredth second (1/100)

Return:

AL=Status (00h on success, FFh on invalid time)

Set the system time.

Note: DD-DOS does not support this function.

9.6.44 AH=2Eh – Set Verify Flag

Input:

AH=2Eh

DL=00h
AL=New state (00h OFF, 01h ON)

Set verify flag.

Note: DD-DOS does not support this function. Verify flag is set by default, and have no effects.

9.6.45 AH=2Fh – Get Disk Transfer Area Address

Input:

AH=2Fh

Return:

ES:BX → Current DTA

Get the address of the current DTA. By default, the DTA is available at PSP:80h.

9.6.46 AH=30h – Get DOS Version

Input:

AH=30h
AL=Type (00h for OEM number, see , otherwise 01h for version)

Return:

AL=Major version
AH=Minor version
BL:CX → 24-bit user serial number (unsupported)
BH=DOS OEM number (AL=00h), or version flag (AL=01h)

Get the current DOS version. By default, DD-DOS returns 5 in AL, 0 in AH, does not use a 24-bit serial number, and return 00h in BH in any cases.

DOS OEM Number	Description
0	IBM
1	Compaq
2	MS Packaged Product
4	AT&T
5	ZDS (Zenith Electronics)

Table 65: DOS OEM Number

9.6.47 AH=31h – Terminate And Stay Resident

Input:

AH=31h
AL=Return code for program
DX=Number of paragraphs (512 bytes blocks) to keep resident

Terminate the current current program with a return code and keep a part of it in memory.

9.6.48 AH=32h – Get Drive Parameter Block For Specified Drive

Input:

AH=32h
DL=Drive number (00h being default, 01h for A:, etc.)

Return:

AL=Status (00h on success, FFh on invalid drive or network drive)
DS:BX → Drive Parameter Block (see DOS Drive Paramater Block Structure)

Get the Drive Parameter Block for a specified drive.

Offset	Size	Description
0	1	Driver number (00h being A:, etc.)
1	1	Unit number within device driver
2	2	Bytes per sector
4	1	Highest sector number within a cluster
5	1	Shift count to convert clusters into sectors
6	2	Number of reserved sectors at beginning of drive
8	1	Number of FAT copies
9	2	Number of root directory entries
11	2	Number of first data sector containing user data
13	2	Highest cluster number (plus 1)
15	1	Number of sectors used by the first FAT entry
16	1	Reserved
17	2	Number of sectors per FAT entry
19	4	Address of current device driver header (see Device Driver Header Structure)
23	1	Media ID byte (see IBM Media IDs)
24	1	00h if disk has been accessed, FFh if not
25	4	Pointer to next Drive Parameter Block Offset is set to FFFFh if it's last block in chain
29	2	Starting cluster for free space search
21	2	Number of free cluster, FFFFh meaning unknown

Table 66: DOS Drive Paramater Block Structure

The Media ID is used to ID drives, such as floppy disk drives and fixed hard disk drives.

Bit	Description
0	If set, 2-sided. Otherwise, not 2-sided
1	If set, 8 sector. Otherwise, not 8 sector
2	If set, removable. Otherwise, fixed
7:3	Must be set

Table 67: Media ID Flags

The possible IDs were defined by IBM.

ID	Description
F8h	Hard disk Double sided 18 sector diskette
F9h	5¼ double-sided, High Density diskette (15 sector) 3½ double-sided, double High Density diskette (9 sector)
FAh	RAM disk, however not all RAM disks use this value
FCh	5¼ single-sided, Double Density diskette (9 sector) 8 inch double-sided, Single Density diskette
FDh	5¼ double-sided, Double Density (9 sector) 8 inch double-sided, Single Density (26 sector) (IBM 3740 format) 128 bytes per sector, soft sector, 4 sectors, cluster, 4 reserved sectors, 2 FATs, 68 directory entries, 77*26*2 sectors.
FEh	5¼ single-sided, Double Density diskette (8 sector) 8 inch single-sided, Single Density diskette (26 sector), (IBM 3740 format) 128 bytes per sector, soft sector, 4 sectors per cluster, 1 reserved sector, 2 FATs. 68 directory entries, 77*26 sectors. 8 inch double-sided, Double Density (8 sector), 1024 bytes per sector, soft sector, 1 sector per cluster, 1 reserved sector, 2 FATs, 192 directory entries, 77*8*2 sectors.
FFh	5¼ double-sided, Double Density diskette (8 sector)

Table 68: IBM Media IDs

9.6.49 AH=33h – Get/Set System Values

Input:

AH=33h
AL=Sub-function (see INT 21h/AH=33h Sub-functions)
DL=Parameter (see INT 21h/AH=33h Sub-function Parameters)

Return:

DL=Result (see INT 21h/AH=33h Return Values)

Get, or set, the CTRL+BREAK checking (for INT 21h calls). This function can also set the boot drive.

Note: DD-DOS always checks for CTRL+BREAK.

AL	Description
00h	Get CTRL+BREAK checking flag
01h	Set CTRL+BREAK checking flag
02h	Set extended CTRL+BREAK checking flag
05h	Get boot drive (where A: is 01h, etc.)
06h	Get original DOS version

Table 69: INT 21h/AH=33h Sub-functions

AL	Description
01h	Set CTRL+BREAK checking, 00h being OFF, and 01h being ON.
05h	Drive number (where 01h is A:, etc.)

Table 70: INT 21h/AH=33h Sub-function Parameters

AL	Description
00h 01h 02h	Get CTRL+BREAK checking, 00h being OFF, and 01h being ON.
05h	Drive number (where 01h is A:, etc.)
06h	DD-DOS always return 5.0. BL=Major version BH=Minor version DL=Revision (always 0) DH=Version flags (always 0)

Table 71: INT 21h/AH=33h Return Values

9.6.50 AH=34h – Get Address Of INDOS Flag

Input:

AH=34h

Return:

ES:BX → INDOS byte

Get the address location of the INDOS flag. The INDOS flag, also known as the DOS Critical Flag, indicates that a DOS call is in-progress if a non-zero value is present.

Note: DD-DOS uses the INDOS pointer in the vdos module.

9.6.51 AH=35h – Get Interrupt Vector

Input:

AH=35h
AL=Interrupt number

Return:

ES:BX → Interrupt handler

Get the address location of an interrupt handler.

Note: DD-DOS processes interrupts statically.

9.6.52 AH=36h – Get Disk Free Space

Input:

AH=36
DL=Drive number (00h being default, 01h for A:, etc.)

Return:

AX=Sectors per cluster (FFFFh on invalid drive)
BX=Number of available clusters
CX=Number of bytes per sector
DX=Number of clusters per drive

Get the amount of free space on disk.

9.6.53 AH=37h – Get/Set Switch Character

Input:

AH=37h
AL=00h for get, 01h for set
DL=New switch character (when AL=01h)

Return:

AL=Status (00h on success, FFh on invalid sub-function)
DL=Current switch character (when AL=00h)

Get, or set, the switch character, using in the command-line interface.

Note: DD-DOS does not support setting the switch character and will always return forward slash '/'.

9.6.54 AH=38h – Get/Set Country Information

Input:

AH=38h
AL=Get country information (00h), or set country information
BX=Country code if AL is greater than FFh
DX=Must be FFFFh to set country information
DS:DX → If AL=00h, specify country data buffer (see DOS Country Information Structure)

Return:

AL=Error code (02h means an invalid country had been specify)
BX=Country code

DS:DX → If AL=00h, country data structure
CF is set on error, cleared otherwise

Get, or set, country information.

Note: DD-DOS does not support setting country information, and will always return USA (01h).

Offset	Size	Description
0	2	Date and time format 00h – MM DD YYYY hh:mm:ss (USA) 01h – DD MM YYYY hh:mm:ss (Europe) 02h – YYYY MM DD hh:mm:ss (Japan)
2	5	ASCIIZ currency symbol, defaults to '$' (DOLLAR-SIGN)
7	2	ASCIIZ thousands separator, defaults to ',' (COMMA)
9	2	ASCIIZ decimal separator, defaults to '.' (PERIOD)
11	2	ASCIIZ date separator, defaults to '-' (DASH)
13	2	ASCIIZ time separator, defaults to ':' (COLUMN)
15	1	Currency symbol format 00h – Symbol leads, without spaces 01h – Symbol follows, without spaces 02h – Symbol leads, one space 03h – Symbol follows, one space 04h – Symbol replaces decimal separator
16	1	Number of digits after the decimal to show
17	1	Time format. If the first bit is set, it indicates a 24-hour clock, 12-hour otherwise
18	4	Address of case map routine
22	2	ASCIIZ data-list separator, defaults to ',' (COMMA)
24	10	Reserved

Table 72: DOS Country Information Structure

9.6.55 AH=39h – Create Sub-directory

Input:

AH=39h
DS:DX → ASCIIZ string path

Return:

AX=Error code if CF is set (see DOS Error Codes)
CF is set on error, cleared otherwise

Create a sub-directory. This function returns an error if the directory already exists, path is not found, the directory is full, or the disk is write-protected.

9.6.56 AH=3Ah – Delete Sub-directory

Input:

AH=3Ah
DS:DX → ASCIIZ string path

Return:

AX=Error code if CF is set (see DOS Error Codes)
CF is set on error, cleared otherwise

Deletes a sub-directory. This function returns an error if the path is not found, the directory is not empty, or the disk is write-protected.

9.6.57 AH=3Bh – Change Current Directory

Input:

AH=3Bh
DS:DX → ASCIIZ string path

Return:

AX=Error code if CF is set (see DOS Error Codes)
CF is set on error, cleared otherwise

Change the current working directory specified in a string in memory. This is equivalent to the chdir or cd commands found in various command-line commands.

9.6.58 AH=3Ch – Create File Using Path

Input:

AH=3Ch
CX=File attributes (see MS-DOS File Attributes)
DS:DX → ASCIIZ string path

Return:

AX=Error code if CF is set (see DOS Error Codes)
CF is set on error, cleared otherwise

Create a file using a string path in memory. If the file already exists, it will be truncated. To create a directory, see AH=39h.

9.6.59 AH=3Dh – Open File Using Path

Input:

AH=3Dh
AL=Access mode (see DOS Files Access Flags)
DS:DX → ASCIIZ string path

Return:

AX=File handle on success, 0000h on error
CF is set on error, cleared otherwise

Open a file handle using a string path in memory. The file handle points at the beginning of the file. A usual access mode is 02h (read+write, b100).

Note: DD-DOS only supports the access mode bits and clears the upper bits.

Bit(s)	Description
0	Read access
1	Write access
2	Read and write access
3	Reserved, always 0
6:4	Sharing mode 000b – compatibility mode (exclusive) 001b – deny others read/write access 010b – deny others write access 011b – deny others read access 100b – full access permitted to all
7	If clear, inheritable. If set, private

Table 73: DOS File Open Mode Attributes

9.6.60 AH=3Eh – Close File Using Handle

Input:

AH=3Eh
BX=File handle

Return:

AX=Error code if CF is set (see DOS Error Codes)
CF is set on error, cleared otherwise

Close a previously opened file handle.

9.6.61 AH=3Fh – Read File Using Handle

Input:

AH=3Fh
BX=File handle
CX=Number of bytes to read
DS:DX → Data buffer

Return:

AX=Number of bytes read on success, error code if CF is set (see DOS Error Codes)
CF is set on error, cleared otherwise

Read the number of specified bytes into buffer located in DS:DX. If AX is unequal to CX, then a partial read was made, since the handle reached the end of file (EOF). This function updates the file position.

Reading from an opened device handle is also possible.

9.6.62 AH=40h – Write To File Using Handle

Input:

AH=40h
BX=File handle
CX=Number of bytes to write
DS:DX → Data buffer

Return:

AX=Number of bytes written on success, error code if CF is set (see DOS Error Codes)
CF is set on error, cleared otherwise

Write the number of specified bytes from buffer located in DS:DX. If AX is unequal to CX, then a partial write was made, since the handle reached the end of file (EOF). This function updates the file position.

Writing to an opened device handle is also possible. A value of 0000h in CX writes no data, and the file is truncated, or extended, to the current file position.

9.6.63 AH=41h – Delete File Using Path

Input:

AH=41h
DS:DX → ASCIIZ string path

Return:

AX=Error code if CF is set (see DOS Error Codes)
CF is set on error, cleared otherwise

Delete a file using a string path in memory.

9.6.64 AH=42h – Set File Position Using Handle

Input:

AH=42h
AL=Origin (00h SEEK_SET, 01h SEEK_CUR, 02h SEEK_END)
BX=File handle
CX:DX=Signed offset from origin for new position

Return:

AX=Error code if CF is set (see DOS Error Codes)
DX:AX=New current file position from beginning of file
CF is set on error, cleared otherwise

Seek to a file position. The SEEK_SET origin starts at the beginning of the file, plus the offset. The SEEK_CUR origin starts at the current file position, plus the offset. The SEEK_END origin starts at the end of the file.

9.6.65 AH=43h – Get/Set File Attributes Using Path

Input:

AH=43h
AL=Get (00h) or set (01h) attribute flag
DS:DX → ASCIIZ string path
CX=Attributes (when AL=01h, see MS-DOS File Attributes)

Return:

AX=Error code if CF is set (see DOS Error Codes)
CX=Attributes (when AL=00h)
CF is set on error, cleared otherwise

Get, or set, file attributes on a file from a string path in memory.

9.6.66 AH=44h – I/O Control For Devices (IOCTL)

This function expands to multiple sub-functions for IOCTL, where the sub-function to invoke is specified in the AL register.

9.6.66.1 IOCTL,0 – GET DEVICE INFORMATION

Input:

AH=44h
AL=00h
BX=File handle (device)

Return:

AX=Error code if CF is set (see DOS Error Codes)
DX=Device information flags (see DOS Block Device Information Flags and DOS Character Device Information Flags)
CF is set on error, cleared otherwise

Get device information from an opened device handle. Bit DX[7] can be used to detect if stdin or stdout is being redirected to, or from, a file.

Bit(s)	Description
5:0	Drive number (00h being A:, etc.)
6	If clear, file has been written
7	If clear, indicates a disk file, otherwise if set, character device
10:8	Reserved
11	If set, the media is not removable
12	If set, indicates a network device
13	Reserved
14	If set, does not update the file time stamp or date stamp
15	If set, the file is remote

Table 74: DOS Block Device Information Flags

Bit(s)	Description
0	If set, indicates input device
1	If set, indicates output device
2	If set, indicates NUL device
3	If set, indicates CLOCK device
4	(read-only) If set, the device uses INT 29h for fast character output
5	If set, indicates binary mode, otherwise if clear, indicates a translation
6	If clear, indicates end of file on input
7	If clear, indicates a disk file, otherwise if set, character device
10:8	Reserved
11	If set, the media is not removable
12	If set, indicates a network device
13	Reserved
14	If set, support IOCTL, via sub-functions 02h and 03h
15	Reserved

Table 75: DOS Character Device Information Flags

9.6.66.2 IOCTL,1 – SET DEVICE INFORMATION

Input:

AH=44h
AL=01h
BX=File handle (character device or file)
DH=Must be cleared (zero)
DL=Device data (see DOS Character Device Information Flags, without read-only tags)

Return:

AX=Error code if CF is set (see DOS Error Codes)
DX=Device information (see DOS Character Device Information Flags)

Set device information from an opened character device handle. This function is usually used to change from binary to translation I/O streams, and can also be used to allow device to expand to words (16-bit) for characters.

Note: DD-DOS does not support setting device information.

9.6.66.3 IOCTL,2 – READ FROM CHARACTER DEVICE

Input:

AH=44h
AL=02h
BX=File handle (device)

CX=Number of bytes to read
DS:DX → Data buffer

Return:

AX=Number of bytes read on success, error code if CF is set (see DOS Error Codes)
CF is set on error, cleared otherwise

Read from a character device.

9.6.66.4 IOCTL,3 – WRITE TO CHARACTER DEVICE

Input:

AH=44h
AL=03h
BX=File handle (device)
CX=Number of bytes to write
DS:DX → Data buffer

Return:

AX=Number of bytes written on success, error code if CF is set (see DOS Error Codes)
CF is set on error, cleared otherwise

Write to a character device.

9.6.66.5 IOCTL,4 – READ FROM BLOCK DEVICE

Input:

AH=44h
AL=04h
BL=Drive number (00h being default, 01h being A:, etc.)
CX=Number of bytes to read
DS:DX → Data buffer

Return:

AX=Number of bytes read on success, error code if CF is set (see DOS Error Codes)
CF is set on error, cleared otherwise

Read from a block device.

9.6.66.6 IOCTL,5 – WRITE TO BLOCK DEVICE

Input:

AH=44h
AL=05h
BL=Drive number (00h being default, 01h being A:, etc.)
CX=Number of bytes to write
DS:DX → Data buffer

Return:

AX=Number of bytes written on success, error code if CF is set (see DOS Error Codes)
CF is set on error, cleared otherwise

Read from the character device.

9.6.66.7 IOCTL,6 – GET INPUT STATUS

Input:

AH=44h
AL=06h
BX=Handle

Return:

AX=Error code if CF is set (see DOS Error Codes)
CF set on error, otherwise cleared

Get the status from an opened handle for input status. Usually used if a file or device is ready for input. It can also be used to determined if EOF has been reached, unless it's caused by INT 21h or 42h.

AL will be set to 00h if EOF has been reached for files, or if the character device is not ready.

AL will be set to FFh if EOF has not been reached for files, or if the character device is ready.

9.6.66.8 IOCTL,7 – GET OUTPUT STATUS

Input:

AH=44h
AL=07h
BX=Handle

Return:

AX=Error code if CF is set (see DOS Error Codes)
CF set on error, otherwise cleared

Get the status from an opened handle for output status. Usually used if a file or device is ready for output.

AL will be set to 00h if EOF has been reached for files, or if the character device is not ready.

AL will be set to FFh if the file or character device is ready.

9.6.66.9 IOCTL,8 – REMOVABLE MEDIA QUERY

Input:

AH=44h
AL=08h
BL=Drive number (00h for default, 01h for A:, etc.)

Return:

AX=00h for removable media, 01h for fixed, error code if CF is set (see DOS Error Codes)
CF set on error, otherwise cleared

Determine if a device is a removable media, like a USB drive. RAM disks are not considered as removable media.

9.6.66.10 IOCTL,9 – LOCAL OR REMOTE DEVICE QUERY USING DRIVE

Input:

AH=44h

AL=09h
BL=Drive number (00h for default, 01h for A:, etc.)

Return:

AX=Error code if CF is set (see DOS Error Codes)
DX=Device attribute word
CF set on error, otherwise cleared

Query the local or remote device from a drive for its status. DX[12] is set if the drive is remote, otherwise cleared for local.

9.6.66.11 IOCTL,10 – LOCAL OR REMOTE DEVICE QUERY USING HANDLE

Input:

AH=44h
AL=0Ah
BL=Handle

Return:

AX=Error code if CF is set (see DOS Error Codes)
DX=Device attribute word
CF set on error, otherwise cleared

Query the local or remote device from a handle for its status. DX[15] is set if the drive is remote, otherwise cleared for local.

9.6.66.12 IOCTL,11 – SET SHARING RETRY COUNT

Input:

AH=44h
AL=0Bh
CX=Pause between retries (defaults to 1)
DX=Number of retries (defaults to 3)

Return:

AX=Error code if CF is set (see DOS Error Codes)
CF set on error, otherwise cleared

Set the retry count for sharing files.

Note: Unsupported in DD-DOS and returns invalid sub-function error.

9.6.66.13 IOCTL,12 – GENERIC I/O OPERATIONS USING HANDLE

Input:

AH=44h
AL=0Ch
BX=Handle
CH=Device type (see IOCTL,12 Device Types)
CL=Sub-function when CH=03h or 05h (see below)
DS:DX → Parameter structure (depends on sub-function)

Return:

AX=Error code if CF is set (see DOS Error Codes)
CF set on error, otherwise cleared

This function groups a few sub-functions whenever CH is set to 03h (CON) or 05h (LPT), otherwise provides generic I/O functions for devices.

CH	Description
00h	Unknown device type
01h	COM (serial port)
03h	CON (console)
05h	LPT (printer port)

Table 76: IOCTL,12 Device Types

To select the code page, set **CL to 45h**. The following structure must be defined and pointed from DS:DX.

Offset	Size	Description
0	2	Length of data
2	2	Code page ID
4	Variable	Character set data array

Table 77: IOCTL,12 Select Code Page Structure

To prepare the start of the code page, set **CL to 4Ch**. The following structure must be defined and pointed from DS:DX.

Offset	Size	Description
0	2	Flags
2	2	Length of remainder for parameter block
4	2	Number of code pages following
6	Variable	Code pages

Table 78: IOCTL,12 Start Code Page Structure

The prepare the end of the code page, set **CL to 4Dh**. The following structure must be defined and pointed from DS:DX.

Offset	Size	Description
0	2	Length of data
2	2	Code page ID

Table 79: IOCTL12 End Code Page Structure

To set the display info, set **CL to 5Fh and CH to 03h**. The following structure must be defined and pointed from DS:DX. Note that DD-DOS does not support graphical modes none other than text.

Offset	Size	Description
0	1	Level (0 for DOS 4.0)
1	1	Reserved
2	2	Length of following data
4	2	Control flags, bits 1 to 15 are reserved Bit 0 – Set for blink, clear for intensity
6	1	Mode type (1 for text, 2 for graphical)
7	1	Reserved
8	2	Colors, 0 for monochrome, otherwise defines bits per pixel
10	2	Width in pixels (columns)
12	2	Height in pixels (rows)
14	2	Columns in characters
16	2	Rows in characters

Table 80: IOCTL,12 Set Display Info Structure

To query the selected code page, set **CL to 6Ah**. The following structure must be defined and pointed from DS:DX.

Offset	Size	Description
0	2	Length of data
2	2	Code page ID
4	Variable	Character set data array

Table 81: IOCTL,12 Query Select Code Page Structure

To query the prepare list, set **CL to 6Bh**. The following structure must be defined and pointed from DS:DX.

Offset	Size	Description
0	2	Length of following data
2	2	Number of hardware code pages
4	Variable	Hardware code pages
Variable	2	Number of prepared code pages
Variable	Variable	Prepared code pages

Table 82: IOCTL,12 Query Prepare List Structure

To get the display info, set **CL to 7Fh and CH to 03h**. The following structure must be defined and pointed from DS:DX.

Offset	Size	Description
0	1	Level (0 for DOS 4.0)
1	1	Reserved
2	2	Length of following data
4	2	Control flags, bits 1 to 15 are reserved Bit 0 – Set for blink, clear for intensity
6	1	Mode type, 1 for text and 2 for graphical
7	1	Reserved
8	2	Colors, 0 for monochrome, otherwise indicates bits per pixels
10	2	Width in pixels (columns)
12	2	Height in pixels (rows)
14	2	Columns in characters
16	2	Rows in characters

Table 83: IOCTL,12 Get Display Info Structure

9.6.66.14 IOCTL,13 – GENERIC I/O OPERATIONS USING DRIVE

Input:

AH=44h
AL=0Dh
BL=Drive number (00h for default, 01h for A:, etc.)
CH=Device type (08h for disk drive, block device)
CL=Sub-function (see below)
DS:DX → Parameter structure (depends on sub-function)

Return:

AX=Error code if CF is set (see DOS Error Codes)
CF set on error, otherwise cleared

This function groups a few sub-functions whenever CH is set to 08h (drive number), otherwise provides generic I/O functions for devices.

To set the device parameters, set **CL to 40h**. To get the device parameters, set **CL to 60h**. The structure must be pointing from DS:DX.

Offset	Size	Description
0	1	Special functions, bits 3-7 are reserved Bit 0 – Set if function uses current BPB, clear for device Bit 1 – Set if function uses track layout field, must be cleared if CL=60h Bit 2 – Set if all sectors in track are the same size (should be)

Table 84: IOCTL,12 Set Device Parameters Structure

To write to a logical device track, set **CL to 41h**. To read from a logical device track, set **CL to 61h**. The structure must be pointing from DS:DX.

Offset	Size	Description
0	1	Reserved, must be 0
1	2	Number of disk head
3	2	Number of disk cylinder
5	2	Number of first sector to read or write
7	2	Number of sectors
9	4	Transfer address

Table 85: IOCTL,13 Logical Device Track

To format a logical device track, set **CL to 42h**.

To verify a logical device track, set **CL to 62h**.

To get the access flag, set **CL to 67h**. The following structure must be pointing from DS:DX.

Offset	Size	Description
0	1	Special function field (reserved, must be zero)
1	1	Disk access flag, non-zero if access is allowed by driver

Table 86: IOCTL,13 Get Access Flags Structure

9.6.66.15 IOCTL,14 – GET LOGICAL DRIVE

Input:

AH=44h
AL=0Eh
BL=Physical drive number (0 for default, 1 for A:, etc.)

Return:

AX=Logical drive number, otherwise error code if CF is set (see DOS Error Codes)
CF set on error, otherwise cleared

Get the logical drive from a physical drive number. Used to determine the last designated drive used to access a drive if more than one logical drive designation applies to a device.

The number is returned in AL. A value of 0 indicates that the drive is accessed by only one drive specifier, otherwise indicates a logical drive number (1 for A:, etc.).

9.6.66.16 IOCTL,15 – SET LOGICAL DRIVE

Input:

AH=44h
AL=0Fh
BL=New drive number (0 for default, 1 for A:, etc.)

Return:

AX=Logical drive number, otherwise error code if CF is set (see DOS Error Codes)
CF set on error, otherwise cleared

Set the logical drive from a drive number. Number returned in AL should be equal to the number entered in BL.

Note: DD-DOS does not support this function.

9.6.67 AH=45h – Duplicate File Handle

Input:
AH=45h
BX=File handle
Return:
AX=New file handle, otherwise error code if CF is set (see DOS Error Codes)
CF set on error, otherwise cleared

Duplicate a file handle. Both file handles move in unison.

9.6.68 AH=46h – Force Duplicate File Handle

Input:
AH=46h
BX=Existing file handle
CX=Second file handle
Return:
AX=Error code if CF is set (see DOS Error Codes)
CF set on error, otherwise cleared

Forcefully duplicate a file handle. If the file handle in CX is already open, it will be closed before copying.

9.6.69 AH=47h – Get Current Directory

Input:
AH=47h
DL=Drive number (0 for default, 1 for A:, etc.)
DS:SI → String buffer
Return:
AX=Error code if CF is set (see DOS Error Codes)
CF set on error, otherwise cleared

Get the working current directory string, returns the directory relative to the root directory. The strong is copied to DS:DI. The leading directory separator ('\') and drive letter are omitted.

Note: DD-DOS limits this string to 64 characters.

9.6.70 AH=48h – Allocate Memory

Input:

AH=48
BX=Number of 512-byte blocks to allocate

Return:

AX=Segment address, otherwise error code if CF is set (see DOS Error Codes)
BX=Size, in 512-byte blocks, of the largest block in memory
CF set on error, otherwise cleared

Allocate memory in paragraphs (512 bytes). The function returns, in AX, the segment of the new allocated memory block. If CF is set and AL is set to 08h, there was not enough memory available to allocate.

There is an overhead of 16 bytes for every memory blocks (MCB).

Note: DD-DOS may allocate memory virtually, but should not allocate additional program memory.

Offset	Size	Description
0	1	Byte signature. 4Dh ('D') if a member of a chain, 5Ah ('Z') for the last entry
1	2	PSP segment address of the MCB owner, otherwise if 0, indicates free, or 8, indicates the block had been allocated by DOS before the first user program loaded
3	2	Number of 512-byte blocks related to this block, excluding this one
5	3	Reserved
8	8	ASCIIZ program name
16	Variable	Allocated memory

Table 87: MS-DOS Memory Block Structure

9.6.71 AH=49h – Free Allocated Memory

Input:

AH=49h
ES=Allocated memory block segment (MCB + 1x 512-byte block)

Return:

AX=Error code if CF is set (see DOS Error Codes)
CF set on error, otherwise cleared

Release memory previously allocated using INT 21h/AH=48h.

9.6.72 AH=4Ah – Modify Allocated Memory Block

Input:

AH=4Ah
BX=New requested block size in 512-byte blocks
ES=Allocated memory block segment (MCB + 1x 512-byte block)

Return:

AX=Error code if CF is set (see DOS Error Codes)

BX=Maximum block size possible, if CF is set and AL=08h
CF set on error, otherwise cleared

Reallocate a memory block previously allocated using INT 21h/AH=48h. If there was not enough memory (CF set and AL=08h), the program has to retry with the value provided in BX.

9.6.73 AH=4Bh – Load And/Or Execute Program

Input:

AH=4Bh
AL=Load type (see MS-DOS EXEC Types)
CX=Mode (when AL=04h only)
DS:DX → ASCIIZ path to executable
ES:BX → Parameter block (see below)

Return:

AX=Error code if CF is set (see DOS Error Codes)
CF set on error, otherwise cleared

Load and/or execute a program on disk. This function creates a PSP.

Note: DD-DOS does not support a load type of 04h.

AL	Description
00h	Load and execute
01h	Load only
03h	Load only (overlay)
04h	(European MS-DOS 4.0) Load and execute in background

Table 88: MS-DOS EXEC Types

When AL is set to 01h, this structure must be present in memory and pointed from ES:BX.

Offset	Size	Description
0	2	Segment of environment or set to 0000h for parent environment
2	4	Pointer to command-line at PSP:80h
6	4	Pointer to default FCB passed at 5Ch
10	4	Pointer to default FCB passed at 6Ch
14	4	Program SS:SP
18	4	Program CS:IP

Table 89: MS-DOS EXEC Type 01 Parameter Structure

When AL is set to 03h, this structure must be present in memory and pointed from ES:BX.

Offset	Size	Description
0	2	Segment of load point for overlay
2	2	Relocation factor for executable overlay
4	4	Pointer to default FCB passed at 5Ch
8	4	Pointer to default FCB passed at 6Ch
12	4	Program SS:SP
16	4	Program CS:IP

Table 90: MS-DOS EXEC Type 03h Parameter Structure

9.6.74 AH=4Ch – Terminate Program With Return Code

Input:

AH=4Ch
AL=Return code (sets ERRORLEVEL)

Terminate the running program with a return code. Recommended by Microsoft to use.

This function flushes all buffers and frees allocated memory, but does not close FCBs.

9.6.75 AH=4Dh – Get Return Code Of Sub-process

Input:

AH=4Dh

Return:

AH=System exit code
AL=Child exit code

Get the return code (ERRORLEVEL) of the sub-process.

9.6.76 AH=4Eh – Find First Matching File Using Path

Input:

AH=4Eh
CX=Search attributes (see MS-DOS File Attributes)
DS:DX → ASCIIZ string path, wildcards included

Return:

AX=Error code if CF is set (see DOS Error Codes)
CF set on error, otherwise cleared

Find the first matching file using a path. The result is stored in the DTA with the following structure.

Note: DD-DOS uses the MS-DOS 5.0 structure.

Offset	Size	Description
0	1	Drive number (0 for default, 1 for A:, etc.)

Offset	Size	Description
1	11	Search template (file name with wildcards)
12	1	Search attributes
13	2	Entry count within directory
15	2	Cluster number of start of parent directory
17	4	Reserved

Table 91: MS-DOS Findfirst Results Structure

9.6.77 AH=4Fh – Find Next Matching File

Input:

AH=4Fh
DS:DX → String path

Return:

AX=Error code if CF is set (see DOS Error Codes)
CF set on error, otherwise cleared

Find the next matching file using the same string used in the INT 21h/AH=4Eh function call in DS:DX.

9.6.78 AH=50h – Set Current Process ID

Input:

AH=50h
BX=Process ID number (processor PSP segment address)

Set a new process ID. The process ID is actually the segment address of a program's PSP, useful for TSR programs.

Note: DD-DOS does not support this function.

9.6.79 AH=51h – Get Current Process ID

Input:

AH=51h

Return:

BX=Process ID

Get the current process ID. The process ID is actually the segment address of a program's PSP, useful for TSR programs.

9.6.80 AH=52h – Get SYSVARS Pointer

Input:

AH=52h

Return:

ES:BS → DOS' list of lists

Get a pointer to lists of pointers for internal parameters, defined below.

Note: DD-DOS does not support CONFIG.SYS.

Offset	Size	Description
-12	2	(SHARE) Retry count
-10	2	(SHARE) Retry delay
-8	4	Pointer to current disk buffer (DTA)
-4	2	Pointer in DOS data segment of unread CON input
-2	2	Pointer to first memory control block (see Memory Control Block Structure)
0	4	Pointer to first Drive Parameter Block (see Drive Parameter Block)
4	4	Pointer to first System File Table (see SFT Structure)
8	4	Pointer to CLOCK$ device driver
12	4	Pointer to CON device driver
16	1	Number of logical drives in system
17	2	Maximum bytes or blocks of any block device
19	2	Number of protected FCBs
21	1	Number of block devices
22	1	Value of LASTDRIVE command in CONFIG.SYS
23	18	NUL device header

Table 92: DOS SYSVARS Pointers List Structure

9.6.81 AH=53h – Generate Drive Parameter Table

Input:

AH=53h
DS:SI → BIOS Parameter Block (see BIOS Parameter Block Structure)
ES:BP → Drive Parameter Block (see Drive Parameter Block Structure)

Return:

ES:BP → Drive Parameter Block

Usually used internally, this function generates a Drive Parameter Table from a BIOS Parameter Block.

Offset	Size	Description
0	2	Sector size in bytes
2	1	Sectors per cluster (size of allocation unit)
3	2	Number of reserved sectors

Offset	Size	Description
5	1	Number of FAT entries on disk
6	2	Number of root directory entries (directory size)
8	2	Number of total sectors; If the partition is bigger than 32M, this is set to zero and field at offset 21 contains the actual count
10	1	Media ID byte (see Media IDs)
11	2	Sectors per FAT entries
13	2	(DOS 3.0+) Sectors per track
15	2	(DOS 3.0+) Number of heads
17	2	(DOS 3.0+) Number of hidden sectors
19	4	(DOS 4.0+) Number of total sectors if offset 8 is zero
23	6	Reserved
29	2	(DOS 4.0+) Number of cylinders
31	1	(DOS 4.0+) Device type
32	2	(DOS 4.0+) Device attributes (see Character Device Attributes and Block Device Attributes)

Table 93: BIOS Parameter Block Structure

9.6.82 AH=54h – Get Verify Flag Setting

Input:

AH=54h

Return:

AL=00h if off, 01h if on

Get the verify setting. The verify flag, if set, verifies data when written on disk.

9.6.83 AH=55h – Create New PSP

Input:

AH=55h
DX=New PSP segment address
SI=Memory size to place in PSP:02h

Create a new PSP in memory. This function is similar to INT 21h/AH=26h, except that the PSP is setup by DOS. All files not marked as inheritable are closed.

9.6.84 AH=56h – Rename File Using Path

Input:

AH=56h

DS:DX → Old ASCIIZ path
ES:DI → New ASCIIZ path

Return:

AX=Error code if CF is set (see DOS Error Codes)
CF set on error, otherwise cleared

Rename a file or directory. This function allows full paths, but not wildcards.

9.6.85 AH=57h/AL=00h – Get File Date And Time Using Handle

Input:

AX=5700h
BX=File handle

Return:

AX=Error code if CF is set (see DOS Error Codes)
CX=File time (see File Time Attributes)
DX=File date (see File Date Attributes)
CF set on error, otherwise cleared

Get the time and date stamps from a file handle. For character devices, MS-DOS returns the date and time of opening.

9.6.86 AH=57h/AL=01h – Set File Date And Time Using Handle

Input:

AX=5701h
BX=File handle
CX=Time (see File Time Attributes)
DX=Date (see File Date Attributes)

Return:

AX=Error code if CF is set (see DOS Error Codes)
CF set on error, otherwise cleared

Set the time and date stamps from a file handle.

9.6.87 AH=58/AH=00h – Get Memory Allocation Strategy

Input:

AX=5800h

Return:

AX=Strategy, otherwise error code if CF is set (see DOS Error Codes)
CF set on error, otherwise cleared

Get the memory allocation strategy. See below for memory allocation strategies.

Note: DD-DOS does not support different memory allocation strategies.

Attribute	Description
00h	Low memory first

Attribute	Description
01h	Low memory is best fit
02h	Low memory is last fit
40h	(DOS 5.0+) High memory first
41h	(DOS 5.0+) High memory is best fit
42h	(DOS 5.0+) High memory is last fit
80h	(DOS 5.0+) First fit, try high memory at first, then low memory
81h	(DOS 5.0+) Best fit, try high memory at first, then low memory
82h	(DOS 5.0+) Last fit, try high memory at first, then low memory

Table 94: DOS Memory Allocation Strategy Attributes

9.6.88 AH=58/AH=01h – Set Memory Allocation Strategy

Input:

AX=5801h
BL=Strategy (see Memory Allocation Strategies)

Return:

AX=Error code if CF is set (see DOS Error Codes)
CF set on error, otherwise cleared

Set the memory allocation strategy.

Note: DD-DOS does not support different memory allocation strategies.

9.6.89 AH=59h – Get Extended Error Code

Input:

AH=59h

Return:

AX=Extended error code (see DOS Error Codes), 0000h if there was no errors
BH=Error class (see DOS Error Classes)
BL=Suggested action (see DOS Error Action Codes)
CH=Locus (see DOS Error Locus)

Get the extended version of the last error code.

Usually, error codes 01h to 12h are returned in AX upon exit from interrupt 21 or 24. The rest can be obtained by issuing this function.

Code	Description
01h	Invalid function number
02h	File not found

Code	Description
03h	Path not found
04h	Too many files opened, no handles left
05h	Access denied
06h	Invalid handle
07h	Memory control blocks destroyed
08h	Insufficient memory
09h	Invalid memory block address
0Ah	Invalid environment
0Bh	Invalid format
0Ch	Invalid access mode (open mode)
0Dh	Invalid data
0Eh	Reserved
0Fh	Invalid drive specified
10h	Attempted to remove current working directory
11h	Not same device
12h	No more files
13h	Attempted to write on a write-protected diskette
14h	Unknown unit
15h	Drive not ready
16h	Unknown command or sub-function
17h	CRC error
18h	Bad request structure length
19h	Seek error
1Ah	Unknown media type
1Bh	Sector not found
1Ch	Printer out of paper
1Dh	Write fault
1Eh	Read fault
1Fh	General failure
20h	Sharing violation

Code	Description
21h	Lock violation
22h	Invalid disk change
23h	FCB unvailable
24h	Sharing buffer overflow
25h	Reserved
26h	Unable to complete file operation
27h-31h	Reserved
32h	Network requests not supported
33h	Remote computer not listening
34h	Duplicate name on network
35h	Network name not found
36h	Network busy
37h	Network device no longer exists
38h	NetBIOS command limit exceeded
39h	Network adapter error
3Ah	Incorrect network response
3Bh	Unexpected network error
3Ch	Incompatible remote adapter
3Dh	Print queue full
3Eh	No space for print file
3Fh	Print file deleted
40h	Network name deleted
41h	Access denied
42h	Network device type is incorrect
43h	Network name not found
44h	Network name limit exceeded
45h	NetBIOS session limit exceeded
46h	Temporary paused
47h	Network request not accepted
48h	Print or disk redirection is paused

Code	Description
49h-4Fh	Reserved
50h	File already exists
51h	Reserved
52h	Could not make directory entry
53h	INT 24h failure
54h	Too many redirections
55h	Duplicate redirection
56h	Invalid password
57h	Invalid parameter
58h	Network device fault
59h	Sub-function not supported by network
5Ah	Required system component missing (not installed)

Table 95: DOS Error Codes

An error class describes which type of error was put in AX, as classification.

Class	Description
01h	Out of resources, out of space, etc.
02h	Temporary situation, e.g. file lock
03h	Authorization, permission denied
04h	Internal error, system detected an internal error
05h	Hardware error
06h	System error, e.g. invalid configuration
07h	Application error
08h	Not found, file or item not found
09h	Bad format, file or item has invalid format
0Ah	Locked, file or item interlocked
0Bh	Media error, includes ECC and CRC errors, or a wrong or bad disk
0Ch	Already exists, file or item already exists
0Dh	Unknown, classification does not exist or could not be classified

Table 96: DOS Error Classes

An action code is a suggested action to take from the last error.

Action Code	Description
01h	Retry, attempt a few times and re-prompt if need be
02h	Delay retry, or retry a few more times after a pause
03h	Re-enter input, re-prompt user if need be
04h	Abort with cleanup, shutdown if need be
05h	Immediate abort, exit immediately with no cleanup
06h	Ignore error
07h	Retry after user fixes issue, requires user intervention

Table 97: DOS Error Action Codes

The locus indicates the position of the error, possible origin of the error.

Error Locus	Description
01h	Unknown
02h	Block device
03h	Network
04h	Serial device
05h	Memory

Table 98: DOS Error Locus

9.6.90 AH=5Ah – Create Temporary File Using Path

Input:

AH=5Ah
CX=File attributes (see DOS File Attributes)
DS:DX → ASCIIZ string path (directory), MUST end with '\' (5Ch)

Return:

AX=File handle, otherwise error code if CF is set (see DOS Error Codes)
DS:DX → Updated ASCIIZ string path to file
CF set on error, otherwise cleared

Create a random, unique, temporary file using a string path in a directory. The the Normal, System, and Hidden file attributes are accepted.

9.6.91 AH=5Bh – Create/Open File Using Path

Input:

AH=5Bh
CX=File attributes (see DOS File Attributes)

DS:DX → ASCIIZ file path

Return:

AX=File handle, otherwise error code if CF is set (see DOS Error Codes)
CF set on error, otherwise cleared

Create, or open, a file. Microsoft recommends using this function. The the Normal, System, and Hidden file attributes are accepted.

9.6.92 AH=5Ch – Lock/Unlock File Access

Input:

AH=5Ch
AL=0 to lock, 1 to unlock
BX=File handle
CX:DX=Region offset
SI:DI=Length

Return:

AX=Error code if CF is set (see DOS Error Codes)
CF set on error, otherwise cleared

Lock, or unlock, an area of a file. The significance is calculated where the most significant register (e.g. CX in CX:DX) is shifted by 4 bits to the left and the least significant register is added to the result.

9.6.93 AH=5Dh – File Sharing functions

This function regroups a multitude of functions relating to sharing files.

Note: DD-DOS does not currently support any of these sub-functions.

9.6.93.1 AL=00H – SERVER FUNCTION CALL

Input:

AX=5D00h
DS:DX → Parameter list (see DOS Parameter List Structure)

Return:

Appropriate for the function being called

Unknown. DPL contains all register values for a call to INT 21h.

Offset	Size	Description
0	2	AX register content
2	2	BX register content
4	2	CX register content
6	2	DX register content
8	2	SI register content

Offset	Size	Description
10	2	DI register content
12	2	DS register content
14	2	ES register content
16	2	Reserved, must be zero
18	2	Computer ID (0 being current system)
20	2	Process ID (PSP segment on specified computer)

Table 99: DOS Parameter List

9.6.93.2 AL=01H – COMMIT ALL FILES FOR COMPUTER

Input:

AX=5D01h

DS:DX → Parameter list (Computer ID and Process ID only, see DOS Parameter List Structure)

Return:

AX=Error code if CF is set (see DOS Error Codes)

CF set on error, otherwise cleared

Flush all buffers for specified computer.

9.6.93.3 AL=02H – SHARE: CLOSE FILE BY NAME

Input:

AX=5D02h

DS:DX → Parameter list (see DOS Parameter List Structure)

(DPL) DS:DX → Full ASCIIZ path

Return:

AX=Error code if CF is set (see DOS Error Codes)

CF set on error, otherwise cleared

This function is used internally to close a shared file by its name.

9.6.93.4 AL=03H – SHARE: CLOSE ALL FILES FOR COMPUTER

Input:

AX=5D03h

DS:DX → Parameter list (Computer ID only, see DOS Parameter List Structure)

Return:

AX=Error code if CF is set (see DOS Error Codes)

CF set on error, otherwise cleared

Close all shared files for a given computer ID.

9.6.93.5 AL=04H – SHARE: CLOSE ALL FILES FOR PROCESS

Input:

AX=5D04h

DS:DX → Parameter list (Computer ID and Process ID only, see DOS Parameter List Structure)

Return:

AX=Error code if CF is set (see DOS Error Codes)

CF set on error, otherwise cleared

Close all shared files for a given process ID.

9.6.93.6 AL=05H – SHARE: GET OPEN FILE LIST ENTRY

Input:

AX=5D05h

DS:DX → Parameter list (see DOS Parameter List Structure)

(DPL) BX=Index of sharing record

(DPL) CX=Index of SFT in the sharing record in SFT list

Return:

AX=0012h on no more files

BX=Network machine number from SFT owner

CX=Number of locks held by SFT owner

ES:DI → ASCIIZ filename

CF set on error (out of range), otherwise cleared

Get the file list entry from a sharing record index.

9.6.93.7 AL=06H – GET ADDRESS OF DOS SWAP DATA AREA

Input:

AX=5D06h

Return:

AX=Error code if CF is set (see DOS Error Codes)

CX=Size of area which must be swapped while in DOS

DX=Size of area which must be always swapped

DS:SI → Non-re-entrant data area (includes all three DOS stacks)

CF set on error, otherwise cleared

Get the address of the DOS swappable data area, including their sizes.

9.6.93.8 AL=07H – GET PRINTER REDIRECTION MODE

Input:

AX=5D07h

Return:

DL=Mode (0 for redirected output is combined, 1 for separate print jobs)

Get the redirection mode for printer.

9.6.93.9 AL=08H – SET PRINTER REDIRECTION MODE

Input:

AX=5D08h
DL=Mode (0 for redirected output is combined, 1 for separate print jobs and start now)

Set the redirection mode for printer.

9.6.93.10 AL=09H – FLUSH REDIRECTED PRINTER OUTPUT

Input:

AX=5D09h

Force the redirected printer output to be printed, and starts a new print job.

9.6.93.11 AL=0AH – SET EXTENDED ERROR INFORMATION

Input:

AH=5D0Ah
DS:DX → Parameter list (see DOS Parameter List Structure)

Set extended error information.

9.6.93.12 AL=0BH – GET DOS SWAP DATA AREAS

Input:

AX=5D0Bh

Return:

AX=Error code if CF is set (see DOS Error Codes)
DS:SI → Swappable data area list (see DOS Swap Data Area List Structure)
CF set on error, otherwise cleared

This function is used internally to get the swappable data areas.

Offset	Size	Description
0	2	Count of data areas
2	Variable	Data area records
n+0	4	Address
n+4	2	Length and type, where bit 15, if set, is always swapped, otherwise swap in DOS

Table 100: DOS Swap Data Area List Structure

9.6.94 AH=5Eh – Networking Functions

This function regroups a multitude of functions related to networking.

Note: DD-DOS does not currently support any of these sub-functions.

9.6.95 AH=5Fh – Network Redirection Functions

This function regroups a multitude of functions related to networking.

Note: DD-DOS does not currently support any of these sub-functions.

9.6.96 AH=60h – Get Fully Qualified Filename

Input:
AH=60h
DS:SI → ASCIIZ string path to evaluate
ES:DI → 128-byte character buffer
Return:
AH=Error code if CF is set
ES:DI → Expanded path
CF set on error, otherwise cleared

Evaluate a path to its full, expanded path. This function does not check if the file exists.

9.6.97 AH=62h – Get Current PSP Address

Input:
AH=62h
Return:
BX=Segment address of current process

Get the PSP address segment, useful for TSR programs.

9.6.98 AH=64h – Set Device Driver Look Ahead

Input:
AH=64h
AL=Switch value

This is an internal function used to set the device driver look ahead strategy. A value of 00h causes DOS to perform the look ahead to the device driver before execution of INT 21h/AH=01h, AH=08h, and AH=0Ah. Any other values indicates to DOS to not look ahead.

9.6.99 AH=65h – Get Extended Country Information

Input:
AH=65h
AL=Sub-function (see DOS Extended Country Information Sub-functions)
BX=Code page (when AL is less than 20h, -1 for current global code page)
CX=Amount of data to return, length of string if AL is 21h or A1h
DX=Country ID (when AL is less than 20h, -1 for current country)
DL=Character to capitalize (when AL is 20h, 23h, or A0h)
DH=Second character if it's a double byte character (when AL is 23h)
ES:DI → Output buffer (when AL is less than 20h, see below)
DS:DX → String to capitalize (when AL is 21h, 22h, A1h, or A2h)
Return:
AX=Error code if CF is set (see DOS Error Codes)

AL=Response (when AL was set to 23h, 0 for NO, 1 for YES, 2 for NEITHER)
DL=Capitalized character (when AL was set to 20h)
CX=Amount of data return
ES:DI → Result table according to sub-function (see below)
CF set on error, otherwise cleared

Get extended country and locale information.

AL	Description
01h	Get extended country information (see Extended Country Information Structure)
02h	Get pointer to character translation table (see Locale Translation Pointer Structure)
04h	Get pointer to filename character translation table (see Locale Translation Pointer Structure)
05h	Get pointer to filename terminator table (see Locale Translation Pointer Structure and DOS File Terminator Table Structure)
06h	Get pointer to collating sequence (see Locale Translation Pointer Structure)
07h	(DOS 4.0+) Get segment of the DCBS vector (see Locale Translation Pointer Structure)
20h	(DOS 4.0+) Country dependent character capitalization
21h	(DOS 4.0+) Country dependent string capitalization
22h	(DOS 4.0+) Country dependent ASCIIZ string capitalization
23h	(DOS 4.0+) Is character country relative
A0h	(DOS 4.0+) Country dependent filename character capitalization
A1h	(DOS 4.0+) Country dependent filename string capitalization
A2h	(DOS 4.0+) Country dependent filename ASCIIZ string capitalization

Table 101: DOS Extended Country Information Sub-functions

Offset	Size	Description
0	1	Sub-function ID (set to 1)
1	2	Size (usually 38 or less)
3	2	Country ID
5	2	Code page
7	2	Date format
9	5	Currency symbol string (null-terminated)
14	2	Thousands separator string (null-terminated)
16	2	Decimal separator string (null-terminated)
18	2	Date separator string (null-terminated)

Offset	Size	Description
20	2	Time separator string (null-terminated)
22	1	Currency symbol location
23	1	Currency decimal location
24	1	Time format
25	4	Extended ASCII map call address
29	2	List separator string (null-terminated)
31	10	Reserved

Table 102: DOS Extended Country Information Structure

Offset	Size	Description
0	1	Sub-function ID (set to 2, 4, 5, 6, or 7 depending on sub-function in AL)
1	4	Pointer to respectful table

Table 103: DOS Locale Translation Pointer Structure

Offset	Size	Description
0	2	Table size
2	7	Unknown
9	1	Length of following data
10	Variable	List of characters for filename terminations, e.g. < > . " / \ [ ] : | + = ; ,

Table 104: DOS File Terminator Table Structure

9.6.100 AH=66h/AL=01h – Get Global Code Page

Input:

AX=6601h

Return:

AX=Error code if CF is set (see DOS Error Codes)
BX=Active code page
CX=System code page
CF set on error, otherwise cleared

Get the global code page.

9.6.101 AH=66h/AL=02h – Set Global Code Page

Input:

AX=6602h
BX=Active code page

CX=System code page

Return:

AX=Error code if CF is set (see DOS Error Codes)
CF set on error, otherwise cleared

Set the global code page.

9.6.102 AH=67h – Set Handle Count

Input:

AH=67h
BX=New maximum

Return:

AX=Error code if CF is set (see DOS Error Codes)
CF set on error, otherwise cleared

Set the new maximum of maximum handles allowed at once for current process. Default being 20. It is not recommended to go past 255.

9.6.103 AH=68h – Flush Buffer Using Handle

Input:

AH=68h
BX=File handle

Return:

AX=Error code if CF is set (see DOS Error Codes)
CF set on error, otherwise cleared

Flush (write) buffers to disk. This function does not update the directory entry and is similar to fflush.

9.6.104 AH=69h/AL=00h – Get Disk Serial Number And Label

Input:

AX=6900h
BL=Drive number (0 being default, 1 for A:, etc.)
DS:DX → Disk information buffer (see DOS Disk Serial/Label Structure)

Return:

AL=Error code if CF is set (see below)
DS:DX → Disk information
CF set on error, otherwise cleared

Internal function. Get the disk information, including the disk serial number and label. This function does not work with network drives.

If CF is set, AL is set to 01h if it detects a network drive and 05h if no extended BPB were found on disk. If CF is cleared, AL is set to 00h if the buffer was filled from the extended BPB.

Note: DD-DOS does not require the setup of a BPB.

(Windows) This function uses GetVolumeInformationA.

Offset	Size	Description
0	2	Info level, always 0000h
2	4	Disk serial number
6	11	Volume label, or "NO NAME " if none
17	8	File-system string, DD-DOS may set custom strings, examples include "FAT12 " 12-bit FAT "FAT16 " 16-bit FAT "CDROM " High-Sierra CD-ROM file-system "CD001 " ISO9660 CD-ROM file-system "CDAUDIO " Audio CD

Table 105: DOS Disk Serial/Label Structure

9.6.105 AH=69h/AL=01h – Set Disk Serial Number And Label

Input:

AX=6901h
BL=Drive number (0 being default, 1 for A:, etc.)
DS:DX → Disk information (see DOS Disk Serial/Label Structure, except file-system field is absent)

Return:

AL=Error code if CF is set (see below)
CF set on error, otherwise cleared

Internal function. Set the disk information. This function does not work with network drives.

If CF is set, AL is set to 01h if it detects a network drive and 05h if no extended BPB were found on disk. If CF is cleared, AL is set to 01h if the extended BPB was updated from buffer

Note: DD-DOS does not support this function.

9.6.106 AH=6Ch – Extended Open/Create File Using Path

Input:

AH=6Ch
BX=Open mode (see DOS Extended File Open Mode Attributes)
CX=File attribute (see DOS File Attributes)
DX=Function control indicator (see)
DS:SI → ASCIIZ string path

Return:

AX=Error code if CF is set (see DOS Error Codes)
CX=Result code (1 on opened, 2 on and opened, 3 on truncated and opened)
CF set on error, otherwise cleared

Create, or open, a file using a path.

Note: DD-DOS does not support sharing modes.

Bit(s)	Description
2:0	Read/write access modes b000 – Read-only b001 – Write-only b010 – Read and write
3	Reserved, zero
6:4	Sharing modes b000 – Compatibility mode b001 – Read/write access b010 – Write access denied b011 – Read access denied b100 – Full access
7	If set, private access; Otherwise, inheritable
12:8	Reserved
13	If set, ignore INT 24h; Otherwise, call INT 24h
14	If set, flush after each write; Otherwise, normal write
15	Reserved

Table 106: DOS Extended File Open Mode Attributes

Bit(s)	Description
3:0	File existence behavior b0000 – If file exists, generate error, and take no action b0001 – If file exists, open file b0010 – If file exists, truncate and open
7:4	Creation behavior b0000 – If file does not exist, generate error, and take no action b0001 – If file does not exist, create file
15:8	Reserved

Table 107: DOS Extended Function Control Attributes

10 Logging

If logging is set, the logger will write messages in a log file (dd-dos.log). Not currently implemented.

Log Level	Alias	Description
0	LOG_SILENCE	No output
1	LOG_CRITICAL	Non-recoverable crashes e.g. Out of bound memory access from the main memory region
2	LOG_ERROR	Operation aborted, program flow may likely be aborted
3	LOG_WARNING	Operation continues, program flow may likely be continued
4	LOG_INFO	Informal messages
5	LOG_DEBUG	Developer messages e.g. Instruction debugging

Table 108: DD-DOS Log Levels

10.1 Bug Codes

All error codes used in DD-DOS have the prefix PANIC_ and covers CLI errors as well. Whenever possible, DD-DOS reuse some of DOS' codes, to avoid redefining duplicate error codes.

DD-DOS, on a non-recoverable error, will display its panic screen. The panic screen includes basic developer information with a 2-byte error code (0000h to FFFFh). The panic screen is not displayed when DD-DOS is still within the command-line.

Error codes will be added with time.

Displayed code	Decimal	Problem Description and Possible Corrective Action
01h	1	(CLI) Invalid or unknown parameter.
02h	2	(CLI) File not found at given location
03h	3	(CLI) File could not be loaded
FFh	255	Unknown error. Default with LOG_CRIT.
DEADh	57005	Manually initiated panic screen.

Table 109: DD-DOS Error Codes

A Glossary

A

ASCIIZ

A null-terminated (zero-terminated) ASCII string.

C

CLI

Command Line Interface, a text-based input interface where commands are executed and the user may specify arguments for the command.

D

DTA

Disk Transfer Area, a data structure MS-DOS uses internally while using I/O related functions.

I

IMR

Interrupt Mask Register, a register within a PIC, the IMR specifies which interrupt can be ignored.

IRQ

Maskable Interrupt, a hardware interrupt that may be ignored with the IMR bit.

P

PIC

Programmable Interrupt Controller, a device that controls the flow and priorities of interrupts.

N

NMI

Non-Maskable Interrupt, a hardware interrupt that the system cannot ignore and can’t be delayed to process.

M

MCB

Memory Control Block, a DOS allocated memory block.

T

TSR

Terminate and Stay Resident, term usually used to denounce TSR programs.

TUI

Text User Interface, where an application runs in a terminal with a graphical fashion.

Examples include Midnight Commander, Microsoft Diagnostics (MSD.EXE), and Rogue.