“Charm can fool you.”

71 Ralf Brown’s Interrupt List

Ralf Brown is a well-known authority for maintaining both documented and undocumented BIOS interrupts, DOS interrupts, memory map and other system-oriented information. Because of him only, the world came to know so many officially undocumented interrupts and system specific information. His work is appreciated throughout the world by thousands of DOS Programmers. The entire Ralf Brown’s Interrupt List is available on CD. The complete list runs up to thousands of pages! Because of space constraint, I provide only a part of Ralf Brown’s Interrupt List. Ralf Brown’s sources are used with his special permission. Many thanks to Dr. Ralf Brown!

71.1 Notations

To save spaces, RBIL (Ralf Brown’s Interrupt List) uses few notations. So we have to understand those notations before using RBIL.

If it is marked "internal" or undocumented, you should check it carefully to make sure it works the same way in your version of the software. Information marked with "???" is known to be incomplete or guesswork.

FLAGS

The use of -> instead of = signifies that the indicated register or register pair contains a pointer to the specified item, rather than the item itself. Register pairs (such as AX:BX) indicate that the item is split across the registers, with the high-order half in the first register.

CATEORIES

The ninth column of the divider line preceding an entry usually contains a classification code (the entry has not been classified if that character is a dash). The codes currently in use are:

A - applications, a - access software (screen readers, etc),
B - BIOS, b - vendor-specific BIOS extensions,
C - CPU-generated, c - caches/spoolers,
D - DOS kernel, d - disk I/O enhancements,
E - DOS extenders, e - electronic mail, F - FAX,
f - file manipulation, G - debuggers/debugging tools, g - games,
H - hardware, h - vendor-specific hardware,
I - IBM workstation/terminal emulators, i - system info/monitoring,
J - Japanese, j - joke programs,
K - keyboard enhancers, k - file/disk compression,
l - shells/command interpreters,
M - mouse/pointing device, m - memory management,
N - network, n - non-traditional input devices,

O - other operating systems,
P - printer enhancements, p - power management,
Q - DESQview/TopView and Quarterdeck programs,
R - remote control/file access, r - runtime support,
S - serial I/O, s - sound/speech,
T - DOS-based task switchers/multitaskers, t - TSR libraries
U - resident utilities, u - emulators,
V - video, v - virus/antivirus,
W - MS Windows,
X - expansion bus BIOSes, x - non-volatile config storage
y - security, * - reserved (and not otherwise classified)

71.2 Interrupt List

71.2.1 Overview

Following is the overall picture about all interrupts.

TITLES

INT 00 - CPU-generated - DIVIDE ERROR
INT 01 - CPU-generated - SINGLE STEP; (80386+) - DEBUGGING EXCEPTIONS
INT 02 - external hardware - NON-MASKABLE INTERRUPT
INT 03 - CPU-generated - BREAKPOINT
INT 04 - CPU-generated - INTO DETECTED OVERFLOW
INT 05 - PRINT SCREEN; CPU-generated (80186+) - BOUND RANGE EXCEEDED
INT 06 - CPU-generated (80286+) - INVALID OPCODE
INT 07 - CPU-generated (80286+) - PROCESSOR EXTENSION NOT AVAILABLE
INT 08 - IRQ0 - SYSTEM TIMER; CPU-generated (80286+)
INT 09 - IRQ1 - KEYBOARD DATA READY; CPU-generated (80286,80386)
INT 0A - IRQ2 - LPT2/EGA,VGA/IRQ9; CPU-generated (80286+)
INT 0B - IRQ3 - SERIAL COMMUNICATIONS (COM2); CPU-generated (80286+)
INT 0C - IRQ4 - SERIAL COMMUNICATIONS (COM1); CPU-generated (80286+)
INT 0D - IRQ5 - FIXED DISK/LPT2/reserved; CPU-generated (80286+)
INT 0E - IRQ6 - DISKETTE CONTROLLER; CPU-generated (80386+)
INT 0F - IRQ7 - PARALLEL PRINTER
INT 10 - VIDEO; CPU-generated (80286+)
INT 11 - BIOS - GET EQUIPMENT LIST; CPU-generated (80486+)
INT 12 - BIOS - GET MEMORY SIZE
INT 13 - DISK
INT 14 - SERIAL
INT 15 - CASSETTE
INT 16 - KEYBOARD
INT 17 - PRINTER
INT 18 - DISKLESS BOOT HOOK (START CASSETTE BASIC)
INT 19 - SYSTEM - BOOTSTRAP LOADER
INT 1A - TIME
INT 1B - KEYBOARD - CONTROL-BREAK HANDLER
INT 1C - TIME - SYSTEM TIMER TICK
INT 1D - SYSTEM DATA - VIDEO PARAMETER TABLES

INT 1E - SYSTEM DATA - DISKETTE PARAMETERS
INT 1F - SYSTEM DATA - 8x8 GRAPHICS FONT
INT 20 - DOS 1+ - TERMINATE PROGRAM
INT 21 - DOS 1+ - Function Calls
INT 22 - DOS 1+ - PROGRAM TERMINATION ADDRESS
INT 23 - DOS 1+ - CONTROL-C/CONTROL-BREAK HANDLER
INT 24 - DOS 1+ - CRITICAL ERROR HANDLER
INT 25 - DOS 1+ - ABSOLUTE DISK READ
INT 26 - DOS 1+ - ABSOLUTE DISK WRITE
INT 27 - DOS 1+ - TERMINATE AND STAY RESIDENT
INT 28 - DOS 2+ - DOS IDLE INTERRUPT
INT 29 - DOS 2+ - FAST CONSOLE OUTPUT
INT 2A - NETBIOS
INT 2B - DOS 2+ - RESERVED
INT 2C - DOS 2+ - RESERVED
INT 2D - DOS 2+ - RESERVED
INT 2E - DOS 2+ - PASS COMMAND TO COMMAND INTERPRETER FOR EXECUTION
INT 2F - Multiplex
INT 30 - (NOT A VECTOR!) - DOS 1+ - FAR JMP instruction
INT 31 - overwritten by CP/M jump instruction in INT 30
INT 32 - (no special use)
INT 33 - MS MOUSE
INT 34 - FLOATING POINT EMULATION - OPCODE D8h
INT 35 - FLOATING POINT EMULATION - OPCODE D9h
INT 36 - FLOATING POINT EMULATION - OPCODE DAh
INT 37 - FLOATING POINT EMULATION - OPCODE DBh
INT 38 - FLOATING POINT EMULATION - OPCODE DCh
INT 39 - FLOATING POINT EMULATION - OPCODE DDh
INT 3A - FLOATING POINT EMULATION - OPCODE DEh
INT 3B - FLOATING POINT EMULATION - OPCODE DFh
INT 3C - FLOATING POINT EMULATION - SEGMENT OVERRIDE
INT 3D - FLOATING POINT EMULATION - STANDALONE FWAIT
INT 3E - FLOATING POINT EMULATION - Borland "SHORTCUT" CALL
INT 3F - Overlay manager interrupt (Microsoft/Borland)
INT 40 - DISKETTE - RELOCATED ROM BIOS DISKETTE HANDLER
INT 41 - SYSTEM DATA - HARD DISK 0 PARAMETER TABLE; CPU - MS Windows
INT 42 - VIDEO - RELOCATED DEFAULT INT 10 VIDEO SERVICES (EGA,VGA)
INT 43 - VIDEO DATA - CHARACTER TABLE (EGA,MCGA,VGA)
INT 44 - VIDEO DATA - CHARACTER FONT (PCjr); Novell NetWare
INT 45 - Z100/Acorn
INT 46 - SYSTEM DATA - HARD DISK 1 DRIVE PARAMETER TABLE
INT 47 - Z100/Acorn/Western Digital/SQL Base
INT 48 - KEYBOARD (PCjr) - Z100/Watstar/Acorn/Western Digital/Compaq
INT 49 - SYSTEM DATA (PCjr) - Z100/TI/Watstar/Acorn/MAGic
INT 4A - SYSTEM - USER ALARM HANDLER
INT 4B - IBM SCSI interface; Virtual DMA Specification (VDS)
INT 4C - Z100/Acorn/TI
INT 4D - Z100

INT 4E - TI/Z100
INT 4F - Common Access Method SCSI
INT 50 - IRQ0 relocated by software
INT 51 - IRQ1 relocated by software
INT 52 - IRQ2 relocated by software
INT 53 - IRQ3 relocated by software
INT 54 - IRQ4 relocated by software
INT 55 - IRQ5 relocated by software
INT 56 - IRQ6 relocated by software
INT 57 - IRQ7 relocated by software
INT 58 - IRQ8/0 relocated by software
INT 59 - IRQ9/1 relocated by software; GSS Computer Graphics Interface
INT 5A - IRQ10/2 relocated by software
INT 5B - IRQ11/3 relocated by software; Network
INT 5C - IRQ12/4 relocated by software; Network Interface
INT 5D - IRQ13/5 relocated by software
INT 5E - IRQ14/6 relocated by software
INT 5F - IRQ15/7 relocated by software; HP 95LX GRAPHICS PRIMITIVES
INT 60 - reserved for user interrupt; multiple purposes
INT 61 - reserved for user interrupt; multiple purposes
INT 62 - reserved for user interrupt; multiple purposes
INT 63 - reserved for user interrupt; multiple purposes
INT 64 - reserved for user interrupt; multiple purposes
INT 65 - reserved for user interrupt; multiple purposes
INT 66 - reserved for user interrupt; multiple purposes
INT 67 - reserved for user interrupt; LIM EMS; multiple purposes
INT 68 - multiple purposes
INT 69 - multiple purposes
INT 6A - multiple purposes
INT 6B - multiple purposes
INT 6C - CONVERTIBLE; DOS 3.2; DECnet DOS network scheduler
INT 6D - VGA - internal
INT 6E - DECnet DOS - DECnet NETWORK PROCESS API
INT 6F - Novell NetWare; 10NET; MS Windows 3.0
INT 70 - IRQ8 - CMOS REAL-TIME CLOCK
INT 71 - IRQ9 - REDIRECTED TO INT 0A BY BIOS
INT 72 - IRQ10 - RESERVED
INT 73 - IRQ11 - RESERVED
INT 74 - IRQ12 - POINTING DEVICE (PS)
INT 75 - IRQ13 - MATH COPROCESSOR EXCEPTION (AT and up)
INT 76 - IRQ14 - HARD DISK CONTROLLER (AT and later)
INT 77 - IRQ15 - RESERVED (AT,PS); POWER CONSERVATION (Compaq)
INT 78 - DOS extenders; multiple purposes
INT 79 - multiple purposes
INT 7A - Novell NetWare; IBM 3270; multiple purposes
INT 7B - multiple purposes
INT 7C - multiple purposes
INT 7D - multiple purposes

INT 7E - RESERVED FOR DIP, Ltd. ROM LIBRARY; multiple purposes
INT 7F - multiple purposes
INT 80 - reserved for BASIC; multiple purposes
INT 81 - reserved for BASIC
INT 82 - reserved for BASIC
INT 83 - reserved for BASIC
INT 84 - reserved for BASIC
INT 85 - reserved for BASIC
INT 86 - IBM ROM BASIC - used while in interpreter; multiple purposes
INT 87 - IBM ROM BASIC - used while in interpreter
INT 88 - IBM ROM BASIC - used while in interpreter; multiple purposes
INT 89 - IBM ROM BASIC - used while in interpreter
INT 8A - IBM ROM BASIC - used while in interpreter
INT 8B - IBM ROM BASIC - used while in interpreter
INT 8C - IBM ROM BASIC - used while in interpreter
INT 8D - IBM ROM BASIC - used while in interpreter
INT 8E - IBM ROM BASIC - used while in interpreter
INT 8F - IBM ROM BASIC - used while in interpreter
INT 90 - IBM ROM BASIC - used while in interpreter
INT 91 - IBM ROM BASIC - used while in interpreter
INT 92 - IBM ROM BASIC - used while in interpreter; multiple purposes
INT 93 - IBM ROM BASIC - used while in interpreter
INT 94 - IBM ROM BASIC - used while in interpreter; multiple purposes
INT 95 - IBM ROM BASIC - used while in interpreter
INT 96 - IBM ROM BASIC - used while in interpreter
INT 97 - IBM ROM BASIC - used while in interpreter
INT 98 - IBM ROM BASIC - used while in interpreter
INT 99 - IBM ROM BASIC - used while in interpreter
INT 9A - IBM ROM BASIC - used while in interpreter
INT 9B - IBM ROM BASIC - used while in interpreter
INT 9C - IBM ROM BASIC - used while in interpreter
INT 9D - IBM ROM BASIC - used while in interpreter
INT 9E - IBM ROM BASIC - used while in interpreter
INT 9F - IBM ROM BASIC - used while in interpreter
INT A0 - IBM ROM BASIC - used while in interpreter
INT A1 - IBM ROM BASIC - used while in interpreter
INT A2 - IBM ROM BASIC - used while in interpreter
INT A3 - IBM ROM BASIC - used while in interpreter
INT A4 - IBM ROM BASIC - used while in interpreter
INT A5 - IBM ROM BASIC - used while in interpreter
INT A6 - IBM ROM BASIC - used while in interpreter
INT A7 - IBM ROM BASIC - used while in interpreter
INT A8 - IBM ROM BASIC - used while in interpreter
INT A9 - IBM ROM BASIC - used while in interpreter
INT AA - IBM ROM BASIC - used while in interpreter
INT AB - IBM ROM BASIC - used while in interpreter
INT AC - IBM ROM BASIC - used while in interpreter
INT AD - IBM ROM BASIC - used while in interpreter

INT AE - IBM ROM BASIC - used while in interpreter
INT AF - IBM ROM BASIC - used while in interpreter
INT B0 - IBM ROM BASIC - used while in interpreter
INT B1 - IBM ROM BASIC - used while in interpreter
INT B2 - IBM ROM BASIC - used while in interpreter
INT B3 - IBM ROM BASIC - used while in interpreter
INT B4 - IBM ROM BASIC - used while in interpreter
INT B5 - IBM ROM BASIC - used while in interpreter
INT B6 - IBM ROM BASIC - used while in interpreter
INT B7 - IBM ROM BASIC - used while in interpreter
INT B8 - IBM ROM BASIC - used while in interpreter
INT B9 - IBM ROM BASIC - used while in interpreter
INT BA - IBM ROM BASIC - used while in interpreter
INT BB - IBM ROM BASIC - used while in interpreter
INT BC - IBM ROM BASIC - used while in interpreter
INT BD - IBM ROM BASIC - used while in interpreter
INT BE - IBM ROM BASIC - used while in interpreter
INT BF - IBM ROM BASIC - used while in interpreter
INT C0 - IBM ROM BASIC - used while in interpreter
INT C1 - IBM ROM BASIC - used while in interpreter
INT C2 - IBM ROM BASIC - used while in interpreter
INT C3 - IBM ROM BASIC - used while in interpreter
INT C4 - IBM ROM BASIC - used while in interpreter
INT C5 - IBM ROM BASIC - used while in interpreter
INT C6 - IBM ROM BASIC - used while in interpreter
INT C7 - IBM ROM BASIC - used while in interpreter
INT C8 - IBM ROM BASIC - used while in interpreter
INT C9 - IBM ROM BASIC - used while in interpreter
INT CA - IBM ROM BASIC - used while in interpreter
INT CB - IBM ROM BASIC - used while in interpreter
INT CC - IBM ROM BASIC - used while in interpreter
INT CD - IBM ROM BASIC - used while in interpreter
INT CE - IBM ROM BASIC - used while in interpreter
INT CF - IBM ROM BASIC - used while in interpreter
INT D0 - IBM ROM BASIC - used while in interpreter
INT D1 - IBM ROM BASIC - used while in interpreter
INT D2 - IBM ROM BASIC - used while in interpreter
INT D3 - IBM ROM BASIC - used while in interpreter
INT D4 - IBM ROM BASIC - used while in interpreter
INT D5 - IBM ROM BASIC - used while in interpreter
INT D6 - IBM ROM BASIC - used while in interpreter
INT D7 - IBM ROM BASIC - used while in interpreter
INT D8 - IBM ROM BASIC - used while in interpreter
INT D9 - IBM ROM BASIC - used while in interpreter
INT DA - IBM ROM BASIC - used while in interpreter
INT DB - IBM ROM BASIC - used while in interpreter
INT DC - IBM ROM BASIC - used while in interpreter
INT DD - IBM ROM BASIC - used while in interpreter

```
INT DE - IBM ROM BASIC - used while in interpreter
INT DF - IBM ROM BASIC - used while in interpreter
INT E0 - IBM ROM BASIC - used while in interpreter; multiple purposes
INT E1 - IBM ROM BASIC - used while in interpreter
INT E2 - IBM ROM BASIC - used while in interpreter
INT E3 - IBM ROM BASIC - used while in interpreter
INT E4 - IBM ROM BASIC - used while in interpreter
INT E5 - IBM ROM BASIC - used while in interpreter
INT E6 - IBM ROM BASIC - used while in interpreter
INT E7 - IBM ROM BASIC - used while in interpreter
INT E8 - IBM ROM BASIC - used while in interpreter
INT E9 - IBM ROM BASIC - used while in interpreter
INT EA - IBM ROM BASIC - used while in interpreter
INT EB - IBM ROM BASIC - used while in interpreter
INT EC - IBM ROM BASIC - used while in interpreter
INT ED - IBM ROM BASIC - used while in interpreter
INT EE - IBM ROM BASIC - used while in interpreter
INT EF - BASIC - ORIGINAL INT 09 VECTOR
INT F0 - BASICA.COM, GWBASIC, compiled BASIC - ORIGINAL INT 08 VECTOR
INT F1 - reserved for user interrupt
INT F2 - reserved for user interrupt
INT F3 - reserved for user interrupt
INT F4 - reserved for user interrupt
INT F5 - reserved for user interrupt
INT F6 - reserved for user interrupt
INT F7 - reserved for user interrupt
INT F8 - reserved for user interrupt
INT F9 - reserved for user interrupt
INT FA - reserved for user interrupt
INT FB - reserved for user interrupt
INT FC - reserved for user interrupt
INT FD - reserved for user interrupt
INT FE - AT/XT286/PS50+ - destroyed by return from protected mode
INT FF - AT/XT286/PS50+ - destroyed by return from protected mode
```

71.2.2 Listing

Because of space constraint, here I provide only a few interrupts that I use much. The reader is however suggested to check out the CD for complete information. As everyone should be aware of the RBIL format, I present here without formatting it!

```
INT 00 C - CPU-generated - DIVIDE ERROR
Desc:   generated if the divisor of a DIV or IDIV instruction is zero or the
          quotient overflows the result register; DX and AX will be unchanged.
Notes:  on an 8086/8088, the return address points to the following instruction
        on an 80286+, the return address points to the divide instruction
        an 8086/8088 will generate this interrupt if the result of a division
```

```
          is 80h (byte) or 8000h (word)
SeeAlso: INT 04,OPCODE "AAD"
--------G-00--------------------------------
INT 00 - Zenith - ROM DEBUGGER
Desc:   invokes the ROM Debugger when at the BIOS level; equivalent to
          pressing Ctrl-Alt-Ins on booting.
Note:   since DOS revectors INT 00, it is necessary to restore this vector to
          its original ROM BIOS value in order to invoke the debugger once DOS
          loads
SeeAlso: INT 03"Columbia"
--------C-01--------------------------------
INT 01 C - CPU-generated - SINGLE STEP
Desc:   generated after each instruction if TF (trap flag) is set; TF is
          cleared on invoking the single-step interrupt handler
Notes:  interrupts are prioritized such that external interrupts are invoked
          after the INT 01 pushes CS:IP/FLAGS and clears TF, but before the
          first instruction of the handler executes
        used by debuggers for single-instruction execution tracing, such as
          MS-DOS DEBUG's T command
SeeAlso: INT 03"CPU"
--------C-01--------------------------------
INT 01 C - CPU-generated (80386+) - DEBUGGING EXCEPTIONS
Desc:   generated by the CPU on various occurrences which may be of interest
          to a debugger program
Note:   events which may trigger the interrupt:
          Instruction address breakpoint fault - will return to execute inst
          Data address breakpoint trap - will return to following instruction
          General detect fault, debug registers in use
          Task-switch breakpoint trap
          undocumented 386/486 opcode F1h - will return to following instruc
SeeAlso: INT 03"CPU"
--------H-02--------------------------------
INT 02 C - external hardware - NON-MASKABLE INTERRUPT
Desc:   generated by the CPU when the input to the NMI pin is asserted
Notes:  return address points to start of interrupted instruction on 80286+
        on the 80286+, further NMIs are disabled until the next IRET
          instruction, but one additional NMI is remembered by the hardware
          and will be serviced after the IRET instruction reenables NMIs
        maskable interrupts may interrupt the NMI handler if interrupts are
          enabled
        although the Intel documentation states that this interrupt is
          typically used for power-failure procedures, it has many other uses
          on IBM-compatible machines:
                Memory parity error: all except Jr, CONV, and some machines
                                without memory parity
                Breakout switch on hardware debuggers
                Coprocessor interrupt: all except Jr and CONV
                Keyboard interrupt: Jr, CONV
```

```
                I/O channel check: CONV, PS50+
                Disk-controller power-on request: CONV
                System suspend: CONV
                Real-time clock: CONV
                System watch-dog timer, time-out interrupt: PS50+
                DMA timer time-out interrupt: PS50+
                Low battery: HP 95LX
                Module pulled: HP 95LX
--------C-08--------------------------------
INT 08 C - CPU-generated (80286+) - DOUBLE EXCEPTION DETECTED
Desc:   called when multiple exceptions occur on one instruction, or an
          exception occurs in an exception handler
Notes:  called in protected mode if an interrupt above the defined limit of
          the interrupt vector table occurs
        return address points at beginning of instruction with errors or the
          beginning of the instruction which was about to execute when the
          external interrupt caused the exception
        if an exception occurs in the double fault handler, the CPU goes into
          SHUTDOWN mode (which circuitry in the PC/AT converts to a reset);
          this "triple fault" is a faster way of returning to real mode on
          many 80286 machines than the standard keyboard controller reset
--------H-09--------------------------------
INT 09 C - IRQ1 - KEYBOARD DATA READY
Desc:   this interrupt is generated when data is received from the keyboard.
          This is normally a scan code (from either a keypress *or* a key
          release), but may also be an ACK or NAK of a command on AT-class
          keyboards.
Notes:  this IRQ may be masked by setting bit 1 on I/O port 21h
        if the BIOS supports an enhanced (101/102-key) keyboard, it calls
          INT 15/AH=4Fh after reading the scan code (see #00006) from the
          keyboard and before further processing; all further processing uses
          the scan code returned from INT 15/AH=4Fh
        the default interrupt handler is at F000h:E987h in 100%-compatible
          BIOSes
        the interrupt handler performs the following actions for certain
          special keystrokes:
            Ctrl-Break     clear keyboard buffer, place word 0000h in buffer,
                           invoke INT 1B, and set flag at 0040h:0071h
            SysReq         invoke INT 15/AH=85h (SysReq is often labeled SysRq)
            Ctrl-Numlock place system in a tight wait loop until next INT 09
            Ctrl-Alt-Del jump to BIOS startup code (either F000h:FFF0h or the
                             destination of the jump at that address)
            Shift-PrtSc    invoke INT 05
            Ctrl-Alt-Plus (HP Vectra) enable keyclick
            Ctrl-Alt-Plus (many clones) set clock speed to high
            Ctrl-Alt-Minus (HP Vectra) reduce keyclick volume
            Ctrl-Alt-Minus (many clones) set clock speed to low
            Ctrl-Alt-SysReq (HP Vectra) generate hard reset
```

Ctrl-Alt-S (many clones) run BIOS setup program
Ctrl-Alt-Esc (many clones) run BIOS setup program
Ctrl-Alt-Ins (many clones) run BIOS setup program
Ctrl-Alt-LeftShift-GrayMinus (some clones) turn off system cache
Ctrl-Alt-LeftShift-GrayPlus (some clones) turn on system cache
DR DOS hooks this interrupt to control the cursor shape (underscore/ half block) for overwrite/insert mode
DR Multiuser DOS hooks this interrupt for cursor shape control and to control whether Ctrl-Alt-Del reboots the current session or the entire system

SeeAlso: INT 05"PRINT SCREEN",INT 0B"HP 95LX",INT 15/AH=4Fh,INT 15/AH=85h
SeeAlso: INT 16/AH=00h,INT 16/AH=10h,INT 1B,INT 2F/AX=A901h,INT 4A/AH=00h"TI"
SeeAlso: INT 51"DESQview",INT 59"DoubleDOS",INT 79"GO32"

(Table 00006)
Values for keyboard make/break (scan) code:

01h	Esc	31h	N		
02h	1 !	32h	M		
03h	2 @	33h	, <	63h	F16
04h	3 #	34h	. >	64h	F17
05h	4 $	35h	/ ?	65h	F18
06h	5 %	36h	Right Shift	66h	F19
07h	6 ^	37h	Grey*	67h	F20
08h	7 &	38h	Alt	68h	F21 (Fn) [*]
09h	8 *	39h	SpaceBar	69h	F22
0Ah	9 (	3Ah	CapsLock	6Ah	F23
0Bh	0)	3Bh	F1	6Bh	F24
0Ch	- _	3Ch	F2	6Ch	--
0Dh	= +	3Dh	F3	6Dh	EraseEOF
0Eh	Backspace	3Eh	F4		
0Fh	Tab	3Fh	F5	6Fh	Copy/Play
10h	Q	40h	F6		
11h	W	41h	F7		
12h	E	42h	F8	72h	CrSel
13h	R	43h	F9	73h	<delta> [*]
14h	T	44h	F10	74h	ExSel
15h	Y	45h	NumLock	75h	--
16h	U	46h	ScrollLock	76h	Clear
17h	I	47h	Home	77h	[Note2] Joyst But1
18h	O	48h	UpArrow	78h	[Note2] Joyst But2
19h	P	49h	PgUp	79h	[Note2] Joyst Right
1Ah	[{	4Ah	Grey-	7Ah	[Note2] Joyst Left
1Bh	] }	4Bh	LeftArrow	7Bh	[Note2] Joyst Up
1Ch	Enter	4Ch	Keypad 5	7Ch	[Note2] Joyst Down
1Dh	Ctrl	4Dh	RightArrow	7Dh	[Note2] right mouse
1Eh	A	4Eh	Grey+	7Eh	[Note2] left mouse
1Fh	S	4Fh	End		
20h	D	50h	DownArrow		

21h	F	51h	PgDn		
22h	G	52h	Ins		
23h	H	53h	Del		
24h	J	54h	SysReq	---non-key codes---	
25h	K	55h	[Note1] F11	00h	kbd buffer full
26h	L	56h	left | (102-key)		
27h	; :	57h	F11	AAh	self-test complete
28h	' "	58h	F12	E0h	prefix code
29h	` ~	59h	[Note1] F15	E1h	prefix code
2Ah	Left Shift	5Ah	PA1	EEh	ECHO
2Bh	\ |	5Bh	F13 (LWin)	F0h	prefix code (key break)
2Ch	Z	5Ch	F14 (RWin)	FAh	ACK
2Dh	X	5Dh	F15 (Menu)	FCh	diag failure (MF-kbd)
2Eh	C			FDh	diag failure (AT-kbd)
2Fh	V			FEh	RESEND
30h	B			FFh	kbd error/buffer full

Notes: scan codes 56h-E1h are only available on the extended (101/102-key) keyboard and Host Connected (122-key) keyboard; scan codes 5Bh-5Dh are only available on the 122-key keyboard and the Microsoft Natural Keyboard; scan codes 5Eh-76h are only available on the 122-key keyboard

in the default configuration, break codes are the make scan codes with the high bit set; make codes 60h,61h,70h, etc. are not available because the corresponding break codes conflict with prefix codes (code 2Ah is available because the self-test result code AAh is only sent on keyboard initialization). An alternate keyboard configuration can be enabled on AT and later systems with enhanced keyboards, in which break codes are the same as make codes, but prefixed with an F0h scan code

prefix code E0h indicates that the following make/break code is for a "gray" duplicate to a key which existed on the original PC keyboard; prefix code E1h indicates that the following make code has no corresponding break code (currently only the Pause key generates no break code)

the Microsoft Natural Keyboard sends make codes 5Bh, 5Ch, and 5Dh (all with an E0h prefix) for the Left Windows, Right Windows, and Menu keys on the bottom row

the European "Cherry G81-3000 SAx/04" keyboard contains contacts for four additional keys, which can be made available by a user modification; the three new keys located directly below the cursor pad's Delete, End, and PgDn keys send make codes 66h-68h (F19-F21); the fourth new key, named <delta>, sends make code 73h

the SysReq key is often labeled SysRq

the "Accord" ergonomic keyboard with optional touchpad (no other identification visible on keyboard or in owner's booklet) has an additional key above the Grey- key marked with a left-pointing triangle and labeled "Fn" in the owner's booklet which returns scan codes E0h 68h on make and E0h E8h on break

```
          the "Preh Commander AT" keyboard with additional F11-F22 keys treats
            F11-F20 as Shift-F1..Shift-F10 and F21/F22 as Ctrl-F1/Ctrl-F2; the
            Eagle PC-2 keyboard with F11-F24 keys treated those additional keys
            in the same way
          [Note1] the "Cherry G80-0777" keyboard has additional F11-F15 keys
            which generate make codes 55h-59h; some other extended keyboards
            generate codes 55h and 56h for F11 and F12, which cannot be managed
            by standard DOS keyboard drivers
          [Note2] the Schneider/Amstrad PC1512 PC keyboards contain extra keys,
            a mouse, and a digital joystick, which are handled like extra keys.
            The joystick's motion scancodes are converted into standard arrow
            keys by the BIOS, and the joystick and mouse button scan codes are
            converted to FFFFh codes in the BIOS keyboard buffer
            (see CMOS 15h"AMSTRAD").
            In addition to the keys listed in the table above, there are
              Del-> (delete forward)        70h
              Enter                         74h
SeeAlso: #00602 at INT 16/AX=6F07h,#03214 at INT 4A/AH=05h
--------H-0A--------------------------------
INT 0A - IRQ2 - ROLAND MPU MIDI INTERFACE
Note:   newer Roland cards and MIDI interfaces by other manufacturers use
            a jumper-selectable IRQ, but software and hardware generally defaults
            to IRQ2
SeeAlso: INT 52"DESQview",INT 5A"DoubleDOS",INT 71,INT 7A"GO32"
--------V-1000------------------------------
INT 10 - VIDEO - SET VIDEO MODE
          AH = 00h
          AL = desired video mode (see #00010)
Return: AL = video mode flag (Phoenix, AMI BIOS)
              20h mode > 7
              30h modes 0-5 and 7
              3Fh mode 6
          AL = CRT controller mode byte (Phoenix 386 BIOS v1.10)
Desc:   specify the display mode for the currently active display adapter
--------V-1001------------------------------
INT 10 - VIDEO - SET TEXT-MODE CURSOR SHAPE
          AH = 01h
          CH = cursor start and options (see #00013)
          CL = bottom scan line containing cursor (bits 0-4)
Return: nothing
Desc:   specify the starting and ending scan lines to be occupied by the
            hardware cursor in text modes
Notes:  buggy on EGA systems--BIOS remaps cursor shape in 43 line modes, but
            returns unmapped cursor shape
          UltraVision scales size to the current font height by assuming 14-line
            monochrome and 8-line color fonts; this call is not valid if cursor
            emulation has been disabled
          applications which wish to change the cursor by programming the
```

```
          hardware directly on EGA or above should call INT 10/AX=1130h or
          read 0040h:0085h first to determine the current font height
        on some adapters, setting the end line greater than the number of lines
          in the font will result in the cursor extending to the top of the
          next character cell on the right
BUG:    AMI 386 BIOS and AST Premier 386 BIOS will lock up the system if AL
          is not equal to the current video mode
SeeAlso: AH=03h,AX=CD05h,AH=12h/BL=34h,#03885

Bitfields for cursor start and options:
Bit(s)  Description      (Table 00013)
 7      should be zero
 6,5    cursor blink
        (00=normal, 01=invisible, 10=erratic, 11=slow)
        (00=normal, other=invisible on EGA/VGA)
 4-0    topmost scan line containing cursor
--------V-1002------------------------------
INT 10 - VIDEO - SET CURSOR POSITION
        AH = 02h
        BH = page number
           0-3 in modes 2&3
           0-7 in modes 0&1
           0 in graphics modes
        DH = row (00h is top)
        DL = column (00h is left)
Return: nothing
SeeAlso: AH=03h,AH=05h,INT 60/DI=030Bh,MEM 0040h:0050h
--------V-1003------------------------------
INT 10 - VIDEO - GET CURSOR POSITION AND SIZE
        AH = 03h
        BH = page number
           0-3 in modes 2&3
           0-7 in modes 0&1
           0 in graphics modes
Return: AX = 0000h (Phoenix BIOS)
        CH = start scan line
        CL = end scan line
        DH = row (00h is top)
        DL = column (00h is left)
Notes:  a separate cursor is maintained for each of up to 8 display pages
        many ROM BIOSes incorrectly return the default size for a color display
          (start 06h, end 07h) when a monochrome display is attached
        With PhysTechSoft's PTS ROM-DOS the BH value is ignored on entry.
SeeAlso: AH=01h,AH=02h,AH=12h/BL=34h,MEM 0040h:0050h,MEM 0040h:0060h
--------V-1004------------------------------
INT 10 - VIDEO - READ LIGHT PEN POSITION (except VGA)
        AH = 04h
Return: AH = light pen trigger flag
```

```
            00h not down/triggered
            01h down/triggered
                  DH,DL = row,column of character light pen is on
                  CH = pixel row (graphics modes 04h-06h)
                  CX = pixel row (graphics modes with >200 rows)
                  BX = pixel column
Desc:   determine the current position and status of the light pen (if
          present)
Notes:  on a CGA, returned column numbers are always multiples of 2 (320-
          column modes) or 4 (640-column modes)
        returned row numbers are only accurate to two lines
--------V-1004-----------------------------------
INT 10 - HUNTER 16 - GET CURSOR ADDRESS
        AH = 04h
        BH = page
Return: DH = row (0..24)
        DL = column (0..79)
        CH = cursor pixel Y-address (0..199)
        CL = cursor pixel X-address (0..639)
Notes:  the Husky Hunter 16 is an 8088-based ruggedized laptop.       Other family
          members are the Husky Hunter, Husky Hunter 16/80, and Husky Hawk.
        pixel coordinates are for the lower left corner of the character cell
          containing the cursor
SeeAlso: AH=60h"HUNTER"
--------V-1005------------------------------
INT 10 - VIDEO -  SELECT ACTIVE DISPLAY PAGE
        AH = 05h
        AL = new page number (00h to number of pages - 1) (see #00010)
Return: nothing
Desc:   specify which of possibly multiple display pages will be visible
Note:   to determine whether the requested page actually exists, use AH=0Fh
          to query the current page after making this call
SeeAlso: AH=0Fh,AH=43h,AH=45h,MEM 0040h:0062h,MEM 0040h:004Eh
--------V-1006------------------------------
INT 10 - VIDEO - SCROLL UP WINDOW
        AH = 06h
        AL = number of lines by which to scroll up (00h = clear entire window)
        BH = attribute used to write blank lines at bottom of window
        CH,CL = row,column of window's upper left corner
        DH,DL = row,column of window's lower right corner
Return: nothing
Note:   affects only the currently active page (see AH=05h)
BUGS:   some implementations (including the original IBM PC) have a bug which
          destroys BP
        the Trident TVGA8900CL (BIOS dated 1992/9/8) clears DS to 0000h when
          scrolling in an SVGA mode (800x600 or higher)
SeeAlso: AH=07h,AH=12h"Tandy 2000",AH=72h,AH=73h,AX=7F07h,INT 50/AX=0014h
--------V-1007------------------------------
```

```
INT 10 - VIDEO - SCROLL DOWN WINDOW
        AH = 07h
        AL = number of lines by which to scroll down (00h=clear entire window)
        BH = attribute used to write blank lines at top of window
        CH,CL = row,column of window's upper left corner
        DH,DL = row,column of window's lower right corner
Return: nothing
Note:   affects only the currently active page (see AH=05h)
BUGS:  some implementations (including the original IBM PC) have a bug which
          destroys BP
        the Trident TVGA8900CL (BIOS dated 1992/9/8) clears DS to 0000h when
          scrolling in an SVGA mode (800x600 or higher)
SeeAlso: AH=06h,AH=12h"Tandy 2000",AH=72h,AH=73h,INT 50/AX=0014h
--------V-1008-----------------------------
INT 10 - VIDEO - READ CHARACTER AND ATTRIBUTE AT CURSOR POSITION
        AH = 08h
        BH = page number (00h to number of pages - 1) (see #00010)
Return: AH = character's attribute (text mode only) (see #00014)
        AH = character's color (Tandy 2000 graphics mode only)
        AL = character
Notes:  for monochrome displays, a foreground of 1 with background 0 is
          underlined
        the blink bit may be reprogrammed to enable intense background colors
          using AX=1003h or by programming the CRT controller
        the foreground intensity bit (3) can be programmed to switch between
          character sets A and B on EGA and VGA cards, thus enabling 512
          simultaneous characters on screen.  In this case the bit's usual
          function (intensity) is regularly turned off.
        in graphics modes, only characters drawn with white foreground pixels
          are matched by the pattern-comparison routine
        on the Tandy 2000, BH=FFh specifies that the current page should be
          used
        because of the IBM BIOS specifications, there may exist some clone
          BIOSes which do not preserve SI or DI; the Novell DOS kernel
          preserves SI, DI, and BP before many INT 10h calls to avoid problems
          due to those registers not being preserved by the BIOS.
BUG:    some IBM PC ROM BIOSes destroy BP when in graphics modes
SeeAlso: AH=09h,AX=1003h,AX=1103h,AH=12h/BL=37h,AX=5001h

Bitfields for character's display attribute:
Bit(s)  Description      (Table 00014)
 7      foreground blink or (alternate) background bright (see also AX=1003h)
 6-4    background color (see #00015)
 3      foreground bright or (alternate) alternate character set (see AX=1103h)
 2-0    foreground color (see #00015)
SeeAlso: #00026

(Table 00015)
```

```
Values for character color:
        Normal          Bright
 000b   black           dark gray
 001b   blue            light blue
 010b   green           light green
 011b   cyan            light cyan
 100b   red             light red
 101b   magenta                 light magenta
 110b   brown           yellow
 111b   light gray      white
--------V-1009-----------------------------
INT 10 - VIDEO - WRITE CHARACTER AND ATTRIBUTE AT CURSOR POSITION
        AH = 09h
        AL = character to display
        BH = page number (00h to number of pages - 1) (see #00010)
           background color in 256-color graphics modes (ET4000)
        BL = attribute (text mode) or color (graphics mode)
           if bit 7 set in <256-color graphics mode, character is XOR'ed
             onto screen
        CX = number of times to write character
Return: nothing
Notes:  all characters are displayed, including CR, LF, and BS
        replication count in CX may produce an unpredictable result in graphics
          modes if it is greater than the number of positions remaining in the
          current row
        With PhysTechSoft's PTS ROM-DOS the BH, BL, and CX values are ignored
          on entry.
SeeAlso: AH=08h,AH=0Ah,AH=4Bh"GRAFIX",INT 17/AH=60h,INT 1F"SYSTEM DATA"
SeeAlso: INT 43"VIDEO DATA",INT 44"VIDEO DATA"
--------V-100B--BH00------------------------
INT 10 - VIDEO - SET BACKGROUND/BORDER COLOR
        AH = 0Bh
        BH = 00h
        BL = background/border color (border only in text modes)
Return: nothing
SeeAlso: AH=0Bh/BH=01h
--------V-100F-----------------------------
INT 10 - VIDEO - GET CURRENT VIDEO MODE
        AH = 0Fh
Return: AH = number of character columns
        AL = display mode (see #00010 at AH=00h)
        BH = active page (see AH=05h)
Notes:  if mode was set with bit 7 set ("no blanking"), the returned mode will
          also have bit 7 set
        EGA, VGA, and UltraVision return either AL=03h (color) or AL=07h
          (monochrome) in all extended-row text modes
        HP 200LX returns AL=07h (monochrome) if mode was set to AL=21h
          and always 80 resp. 40 columns in all text modes regardless of
```

```
          current zoom setting (see AH=D0h)
        when using a Hercules Graphics Card, additional checks are necessary:
           mode 05h: if WORD 0040h:0063h is 03B4h, may be in graphics page 1
                  (as set by DOSSHELL and other Microsoft software)
           mode 06h: if WORD 0040h:0063h is 03B4h, may be in graphics page 0
                  (as set by DOSSHELL and other Microsoft software)
           mode 07h: if BYTE 0040h:0065h bit 1 is set, Hercules card is in
                  graphics mode, with bit 7 indicating the page (mode set by
                  Hercules driver for Borland Turbo C)
        the Tandy 2000 BIOS is only documented as returning AL, not AH or BH
SeeAlso: AH=00h,AH=05h,AX=10F2h,AX=1130h,AX=CD04h,MEM 0040h:004Ah
--------V-1010------------------------------
INT 10 - Tandy 2000 - VIDEO - GET/SET CHARACTER FONTS
        AH = 10h
        AL = control value
           bit 0: set character set instead of reading it
           bit 1: high 128 characters instead of low 128 characters
        ES:BX -> new character set if AL bit 0 set
Return: ES:BX -> current character set if AL bit 0 clear on entry
Notes:  this interrupt is identical to INT 52 on Tandy 2000
        the character set consists of 16 bytes for each of the 128 characters,
          where each of the 16 bytes describes the pixels in one scan line,
          most significant bit leftmost
SeeAlso: AH=00h,AH=0Bh/BH=02h,AH=11h"Tandy 2000",AH=12h"Tandy 2000"
SeeAlso: INT 52"Tandy 2000"
--------V-101104----------------------------
INT 10 - VIDEO - TEXT-MODE CHARGEN - LOAD ROM 8x16 CHARACTER SET (VGA)
        AX = 1104h
        BL = block to load
Return: nothing
Notes:  (see AX=1100h)
SeeAlso: AX=1100h,AX=1101h,AX=1102h,AX=1103h,AX=1114h,AH=1Bh,AX=CD10h
SeeAlso: MEM 0040h:0084h
Index:  text mode;font|text mode;screen rows
--------J-1018------------------------------
INT 10 - VIDEO - DOS/V - GET/SET FONT PATTERN
        AH = 18h
        AL = subfunction
           00h get font pattern
           01h set font pattern
        BX = 0000h
        CL = character size in bytes (01h,02h)
        CH = 00h
        DH = character width in pixels
        DL = character height in pixels
        ES:DI -> buffer for/containing font image
Return: AL = status (00h successful, else error)
        ES:DI buffer filled for function 00h if successful
```

```
Note:   the supported font sizes are 8x16 single-byte, 8x19 single-byte,
          16x16 double-byte, and 24x24 double-byte
SeeAlso: AH=19h,INT 16/AH=14h
--------V-101E08----------------------------
INT 10 - VIDEO - FLAT-PANEL - CONTRAST SETTING
        AX = 1E08h
        BH = function
           bit 7: =1 set contrast control, =0 query contrast
           bit 6: use standard contrast
           bits 5-0: reserved (0)
        ---if BH bits 7,6=10---
        BL = contrast (00h = minimum, FFh = maximum)
Return: AL = 1Eh if function supported
           BH = results
                 bit 7: query/set (copied from input)
                 bit 6: standard/custom (copied from input)
                 bits 5-2: reserved (0)
                 bit 1: software contrast control is supported
                 bit 0: set operation was succesful (always clear on get)
           BL = contrast (00h = minimum, FFh = maximum)
Note:   this function operates independently of AX=1E06h
SeeAlso: AX=1E00h,AX=1E06h,AX=1E07h
--------V-104F00----------------------------
INT 10 - VESA SuperVGA BIOS (VBE) - GET SuperVGA INFORMATION
        AX = 4F00h
        ES:DI -> buffer for SuperVGA information (see #00077)
Return: AL = 4Fh if function supported
        AH = status
           00h successful
                 ES:DI buffer filled
           01h failed
           ---VBE v2.0---
           02h function not supported by current hardware configuration
           03h function invalid in current video mode
Desc:   determine whether VESA BIOS extensions are present and the capabilities
          supported by the display adapter
SeeAlso: AX=4E00h,AX=4F01h,AX=7F00h"SOLLEX",AX=A00Ch
Index:  installation check;VESA SuperVGA

Format of SuperVGA information:
Offset  Size     Description      (Table 00077)
 00h  4 BYTEs  (ret) signature ("VESA")
                 (call) VESA 2.0 request signature ("VBE2"), required to receive
                   version 2.0 info
 04h     WORD  VESA version number (one-digit minor version -- 0102h = v1.2)
 06h     DWORD          pointer to OEM name
                 "761295520" for ATI
 0Ah     DWORD          capabilities flags (see #00078)
```

```
 0Eh     DWORD          pointer to list of supported VESA and OEM video modes
                  (list of words terminated with FFFFh)
 12h     WORD  total amount of video memory in 64K blocks
---VBE v1.x ---
 14h 236 BYTEsreserved
---VBE v2.0 ---
 14h     WORD  OEM software version (BCD, high byte = major, low byte = minor)
 16h     DWORD          pointer to vendor name
 1Ah     DWORD          pointer to product name
 1Eh     DWORD          pointer to product revision string
 22h     WORD  (if capabilities bit 3 set) VBE/AF version (BCD)
                  0100h for v1.0P
 24h     DWORD          (if capabilities bit 3 set) pointer to list of supported
                    accelerated video modes (list of words terminated with FFFFh)
 28h 216 BYTEsreserved for VBE implementation
100h 256 BYTEs          OEM scratchpad (for OEM strings, etc.)
Notes:  the list of supported video modes is stored in the reserved portion of
          the SuperVGA information record by some implementations, and it may
          thus be necessary to either copy the mode list or use a different
          buffer for all subsequent VESA calls
        not all of the video modes in the list of mode numbers may be
          supported, e.g. if they require more memory than currently installed
          or are not supported by the attached monitor. Check any mode you
          intend to use through AX=4F01h first.
        the 1.1 VESA document specifies 242 reserved bytes at the end, so the
          buffer should be 262 bytes to ensure that it is not overrun; for
          v2.0, the buffer should be 512 bytes
        the S3 specific video modes will most likely follow the FFFFh
          terminator at the end of the standard modes. A search must then
          be made to find them, FFFFh will also terminate this second list
        in some cases, only a "stub" VBE may be present, supporting only
          AX=4F00h; this case may be assumed if the list of supported video
          modes is empty (consisting of a single word of FFFFh)

Bitfields for VESA capabilities:
Bit(s)  Description      (Table 00078)
 0      DAC can be switched into 8-bit mode
 1      non-VGA controller
 2      programmed DAC with blank bit (i.e. only during blanking interval)
 3      (VBE v3.0) controller supports hardware stereoscopic signalling
 3      controller supports VBE/AF v1.0P extensions
 4      (VBE v3.0) if bit 3 set:
           =0 stereo signalling via external VESA stereo connector
           =1 stereo signalling via VESA EVC connector
 4      (VBE/AF v1.0P) must call EnableDirectAccess to access framebuffer
 5      (VBE/AF v1.0P) controller supports hardware mouse cursor
 6      (VBE/AF v1.0P) controller supports hardware clipping
 7      (VBE/AF v1.0P) controller supports transparent BitBLT
```

```
 8-31   reserved (0)
SeeAlso: #00077,AX=4F09h
--------V-104F01----------------------------
INT 10 - VESA SuperVGA BIOS - GET SuperVGA MODE INFORMATION
        AX = 4F01h
        CX = SuperVGA video mode (see #04082 for bitfields)
        ES:DI -> 256-byte buffer for mode information (see #00079)
Return: AL = 4Fh if function supported
        AH = status
           00h successful
                ES:DI buffer filled
           01h failed
Desc:   determine the attributes of the specified video mode
SeeAlso: AX=4F00h,AX=4F02h

Bitfields for VESA/VBE video mode number:
Bit(s)  Description      (Table 04082)
 15     preserve display memory on mode change
 14     (VBE v2.0+) use linear (flat) frame buffer
 13     (VBE/AF 1.0P) VBE/AF initializes accelerator hardware
 12     reserved for VBE/AF
 11     (VBE v3.0) user user-specified CRTC refresh rate values
 10-9   reserved for future expansion
 8-0    video mode number (0xxh are non-VESA modes, 1xxh are VESA-defined)

Format of VESA SuperVGA mode information:
Offset  Size    Description      (Table 00079)
 00h    WORD    mode attributes (see #00080)
 02h    BYTE    window attributes, window A (see #00081)
 03h    BYTE    window attributes, window B (see #00081)
 04h    WORD    window granularity in KB
 06h    WORD    window size in KB
 08h    WORD    start segment of window A (0000h if not supported)
 0Ah    WORD    start segment of window B (0000h if not supported)
 0Ch    DWORD           -> FAR window positioning function (equivalent to AX=4F05h)
 10h    WORD    bytes per scan line
---remainder is optional for VESA modes in v1.0/1.1, needed for OEM modes---
 12h    WORD    width in pixels (graphics) or characters (text)
 14h    WORD    height in pixels (graphics) or characters (text)
 16h    BYTE    width of character cell in pixels
 17h    BYTE    height of character cell in pixels
 18h    BYTE    number of memory planes
 19h    BYTE    number of bits per pixel
 1Ah    BYTE    number of banks
 1Bh    BYTE    memory model type (see #00082)
 1Ch    BYTE    size of bank in KB
 1Dh    BYTE    number of image pages (less one) that will fit in video RAM
 1Eh    BYTE    reserved (00h for VBE 1.0-2.0, 01h for VBE 3.0)
```

```
---VBE v1.2+ ---
 1Fh    BYTE    red mask size
 20h    BYTE    red field position
 21h    BYTE    green mask size
 22h    BYTE    green field size
 23h    BYTE    blue mask size
 24h    BYTE    blue field size
 25h    BYTE    reserved mask size
 26h    BYTE    reserved mask position
 27h    BYTE    direct color mode info
                bit 0: color ramp is programmable
                bit 1: bytes in reserved field may be used by application
---VBE v2.0+ ---
 28h    DWORD          physical address of linear video buffer
 2Ch    DWORD          pointer to start of offscreen memory
 30h    WORD  KB of offscreen memory
---VBE v3.0 ---
 32h    WORD  bytes per scan line in linear modes
 34h    BYTE    number of images (less one) for banked video modes
 35h    BYTE    number of images (less one) for linear video modes
 36h    BYTE    linear modes: size of direct color red mask (in bits)
 37h    BYTE    linear modes: bit position of red mask LSB (e.g. shift count)
 38h    BYTE    linear modes: size of direct color green mask (in bits)
 39h    BYTE    linear modes: bit position of green mask LSB (e.g. shift count)
 3Ah    BYTE    linear modes: size of direct color blue mask (in bits)
 3Bh    BYTE    linear modes: bit position of blue mask LSB (e.g. shift count)
 3Ch    BYTE    linear modes: size of direct color reserved mask (in bits)
 3Dh    BYTE    linear modes: bit position of reserved mask LSB
 3Eh    DWORD          maximum pixel clock for graphics video mode, in Hz
 42h 190 BYTEsreserved (0)
Note:   while VBE 1.1 and higher will zero out all unused bytes of the buffer,
          v1.0 did not, so applications that want to be backward compatible
          should clear the buffer before calling

Bitfields for VESA SuperVGA mode attributes:
Bit(s)  Description      (Table 00080)
 0      mode supported by present hardware configuration
 1      optional information available (must be =1 for VBE v1.2+)
 2      BIOS output supported
 3      set if color, clear if monochrome
 4      set if graphics mode, clear if text mode
---VBE v2.0+ ---
 5      mode is not VGA-compatible
 6      bank-switched mode not supported
 7      linear framebuffer mode supported
 8      double-scan mode available (e.g. 320x200 and 320x240)
---VBE v3.0 ---
 9      interlaced mode available
```

```
 10      hardware supports triple buffering
 11      hardware supports stereoscopic display
 12      dual display start address support
 13-15   reserved
---VBE/AF v1.0P---
 9       application must call EnableDirectAccess before calling bank-switching
           functions
SeeAlso: #00079

Bitfields for VESA SuperVGA window attributes:
Bit(s)   Description      (Table 00081)
 0       exists
 1       readable
 2       writable
 3-7     reserved
SeeAlso: #00079

(Table 00082)
Values for VESA SuperVGA memory model type:
 00h     text
 01h     CGA graphics
 02h     HGC graphics
 03h     16-color (EGA) graphics
 04h     packed pixel graphics
 05h     "sequ 256" (non-chain 4) graphics
 06h     direct color (HiColor, 24-bit color)
 07h     YUV (luminance-chrominance, also called YIQ)
 08h-0Fh reserved for VESA
 10h-FFh OEM memory models
SeeAlso: #00079
--------V-104F02----------------------------
INT 10 - VESA SuperVGA BIOS - SET SuperVGA VIDEO MODE
         AX = 4F02h
         BX = new video mode (see #04082,#00083,#00084)
         ES:DI -> (VBE 3.0+) CRTC information block, bit mode bit 11 set
                    (see #04083)
Return: AL = 4Fh if function supported
         AH = status
             00h successful
             01h failed
Notes:  bit 13 may only be set if the video mode is present in the list of
           accelerated video modes returned by AX=4F00h
         if the DAC supports both 8 bits per primary color and 6 bits, it will
           be reset to 6 bits after a mode set; use AX=4F08h to restore 8 bits
SeeAlso: AX=4E03h,AX=4F00h,AX=4F01h,AX=4F03h,AX=4F08h

(Table 00083)
Values for VESA video mode:
```

```
 00h-FFh OEM video modes (see #00010 at AH=00h)
 100h   640x400x256
 101h   640x480x256
 102h   800x600x16
 103h   800x600x256
 104h   1024x768x16
 105h   1024x768x256
 106h   1280x1024x16
 107h   1280x1024x256
 108h   80x60 text
 109h   132x25 text
 10Ah   132x43 text
 10Bh   132x50 text
 10Ch   132x60 text
---VBE v1.2+ ---
 10Dh   320x200x32K
 10Eh   320x200x64K
 10Fh   320x200x16M
 110h   640x480x32K
 111h   640x480x64K
 112h   640x480x16M
 113h   800x600x32K
 114h   800x600x64K
 115h   800x600x16M
 116h   1024x768x32K
 117h   1024x768x64K
 118h   1024x768x16M
 119h   1280x1024x32K (1:5:5:5)
 11Ah   1280x1024x64K (5:6:5)
 11Bh   1280x1024x16M
---VBE 2.0+ ---
 120h   1600x1200x256
 121h   1600x1200x32K
 122h   1600x1200x64K
81FFh   special full-memory access mode
Notes:  the special mode 81FFh preserves the contents of the video memory and
          gives access to all of the memory; VESA recommends that the special
          mode be a packed-pixel mode.        For VBE 2.0+, it is required that the
          VBE implement the mode, but not place it in the list of available
          modes (mode information for this mode can be queried directly,
          however).
        as of VBE 2.0, VESA will no longer define video mode numbers
SeeAlso: #00010,#00011,#00084,#00191
Index:  video modes;VESA

(Table 00084)
Values for S3 OEM video mode:
 201h   640x480x256
```

```
 202h   800x600x16
 203h   800x600x256
 204h   1024x768x16
 205h   1024x768x256
 206h   1280x960x16
 207h   1152x864x256 (Diamond Stealth 64)
 208h   1280x1024x16
 209h   1152x864x32K
 20Ah   1152x864x64K (Diamond Stealth 64)
 20Bh   1152x864x4G
 211h   640x480x64K (Diamond Stealth 24)
 211h   640x400x4G  (Diamond Stealth64 Video / Stealth64 Graphics)
 212h   640x480x16M (Diamond Stealth 24)
 301h   640x480x32K
Note:   these modes are only available on video cards using S3's VESA driver
SeeAlso: #00083,#00191,#00732 at INT 1A/AX=B102h
Index:  video modes;S3

Format of VESA VBE CRTC Information Block:
Offset  Size     Description      (Table 04083)
 00h    WORD     total number of pixels horizontally
 02h    WORD     horizontal sync start (in pixels)
 04h    WORD     horizontal sync end (in pixels)
 06h    WORD     total number of scan lines
 08h    WORD     vertical sync start (in scan lines)
 0Ah    WORD     vertical sync end (in scan lines)
 0Ch    BYTE     flags (see #04084)
 0Dh    DWORD          pixel clock, in Hz
 11h    WORD     refresh rate, in 0.01 Hz units
                 this field MUST be set to pixel_clock / (HTotal * VTotal),
                   even though it may not actually be used by the VBE
                   implementation
 13h 40 BYTEs    reserved

Bitfields for VESA VBE CRTC Information Block flags:
Bit(s)  Description      (Table 04084)
 0      enable double scanning
 1      enable interlacing
 2      horizontal sync polarity (0 positive, 1 negative)
 3      vertical sync polarity (0 positive, 1 negative)
SeeAlso: #04083
--------V-104F03----------------------------
INT 10 - VESA SuperVGA BIOS - GET CURRENT VIDEO MODE
        AX = 4F03h
Return: AL = 4Fh if function supported
        AH = status
           00h successful
                BX = video mode (see #00083,#00084)
```

```
                    bit 13: VBE/AF v1.0P accelerated video mode
                    bit 14: linear frame buffer enabled (VBE v2.0+)
                    bit 15: don't clear video memory
            01h failed
SeeAlso: AH=0Fh,AX=4E04h,AX=4F02h
--------V-104F04----------------------------
INT 10 - VESA SuperVGA BIOS - SAVE/RESTORE SuperVGA VIDEO STATE
        AX = 4F04h
        DL = subfunction
            00h get state buffer size
                Return: BX = number of 64-byte blocks needed
            01h save video states
                ES:BX -> buffer
            02h restore video states
                ES:BX -> buffer
        CX = states to save/restore (see #00085)
Return: AL = 4Fh if function supported
        AH = status
            00h successful
            01h failed
SeeAlso: AH=1Ch,AX=5F90h,AX=5FA0h

Bitfields for VESA SuperVGA states to save/restore:
Bit(s)  Description     (Table 00085)
 0      video hardware state
 1      video BIOS data state
 2      video DAC state
 3      SuperVGA register state
SeeAlso: #00048,#00186
--------s-104F13BX0002-----------------------
INT 10 - VESA VBE/AI (Audio Interface) - QUERY DEVICE
        AX = 4F13h
        BX = 0002h
        CX = handle
        DX = query
            0001h return length of GeneralDeviceClass
            0002h return copy   of GeneralDeviceClass (see #00112)
            0003h return length of Volume Info Structure
            0004h return copy   of Volume Info Structure (see #00122)
            0005h return length of Volume Services Structure
            0006h return copy   of Volume Services Structure (see #00124)
            0007h-000Fh reserved
            0010h-FFFFh device-specific
        SI:DI -> buffer (functions 0002h,0004h,0006h)
Return: AL = 4Fh if function supported
            AH = status
                00h successful
                    SI:DI = length (functions 1,3,5)
```

```
                    SI:DI buffer filled (functions 2,4,6)
                 01h failed
Note:   functions 0003h to 0006h are only supported for the Volume device

Format of GeneralDeviceClass structure:
Offset  Size     Description      (Table 00112)
 00h    4 BYTEs          name of the structure ("GENI")
 04h    DWORD            structure length
 08h    WORD   type of device (1=Wave, 2=MIDI)
 0Ah    WORD   version of VESA driver support (0100h for 1.00)
 10h    var    for CX=handle for Wave device:
                    Wave Info structure (see #00113)
                    some bytes ???
                for CX=handle for MIDI device:
                    MIDI Info Structure (see #00118)
                    first 8 bytes of MIDI Service Structure ???
SeeAlso: #00122,#00124

Format of WAVE Info Structure:
Offset  Size     Description      (Table 00113)
 00h    4 BYTEs          name of the structure ("WAVI")
 04h    DWORD            structure length [0000007Eh]
 08h    DWORD            driver software version [00000003h]
 0Ch 32 BYTEs   vendor name, etc. (ASCIZ string)
 2Ch 32 BYTEs   vendor product name
 4Ch 32 BYTEs   vendor chip/hardware description
 6Ch    BYTE   installed board number
 6Dh    3 BYTEs          unused data
 70h    DWORD            feature bits (see #00114)
 74h    WORD   user determined preference field
 76h    WORD   memory required for driver use [0200h]
 78h    WORD   number of timer tick callbacks per second [0000h]
 7Ah    WORD   channels: 1 = mono, 2 = stereo
                stereo is assumed to be interleaved data
 7Ch    WORD   bitfield of max sample sizes (see #00115)
SeeAlso: #00118

Bitfields for Wave feature bits:
Bit(s)  Description      (Table 00114)
 0      8000hz Mono Playback
 1      8000hz Mono Record
 2      8000hz Stereo Record
 3      8000hz Stereo Playback
 4      8000hz Full Duplex Play/Record
 5      11025hz Mono Playback
 6      11025hz Mono Record
 7      11025hz Stereo Record
 8      11025hz Stereo Playback
```

```
 9       11025hz Full Duplex Play/Record
 10      22050hz Mono Playback
 11      22050hz Mono Record
 12      22050hz Stereo Record
 13      22050hz Stereo Playback
 14      22050hz Full Duplex Play/Record
 15      44100hz Mono Playback
 16      44100hz Mono Record
 17      44100hz Stereo Record
 18      44100hz Stereo Playback
 19      44100hz Full Duplex Play/Record
 20-26   reserved (0)
 27      driver must pre-handle the data
 28      Variable Sample mono   playback
 29      Variable Sample stereo playback
 30      Variable Sample mono   record
 31      Variable Sample stereo record

(Table 00115)
Values for Sample data size:
 01h     8bit play
 02h     16bit play
 10h     8bit record
 20h     16bit record

Format of WAVE Audio Services structure:
Offset   Size     Description      (Table 00116)
 00h     4 BYTEs          name of the structure
 04h     DWORD            structure length
 08h 16 BYTEs  for future expansion
---entry points (details???)---
 18h     DWORD            DeviceCheck
                  11h compression (see also #00117)
                  12h driver state
                  13h get current pos
                  14h sample rate
                  15h set preference
                  16h get DMA,IRQ
                  17h get IO address
                  18h get mem address
                  19h get mem free
                  1Ah full duplex
                  1Bh get block size
                  1Ch get PCM format
                  1Dh enable PCM format
                  80h-.. vendors can add DevChks above 0x80
 1Ch     DWORD            PCMInfo
 20h     DWORD            PlayBlock
```

24h	DWORD	PlayCont
28h	DWORD	RecordBlock
2Ch	DWORD	RecordCont
30h	DWORD	PauseIO
34h	DWORD	ResumeIO
38h	DWORD	StopIO
3Ch	DWORD	WavePrepare
40h	DWORD	WaveRegister
44h	DWORD	GetLastError

01h unsupported feature/function
02h bad sample rate
03h bad block length
04h bad block address
05h app. missed an IRQ
06h don't understand the PCM size/format
80h-.. vendors specific errors

48h	DWORD	TimerTick
4Ch	DWORD	ApplPSyncCB: CallBack: play filled in by the app
50h	DWORD	ApplRSyncCB: CallBack: rec filled in by the app

SeeAlso: #00120,#00124

(Table 00117)
Values for type of compression:

01h	IMA play
02h	ALAW play
03h	ULAW play
11h	IMA record
12h	ALAW record
13h	ULAW record

Format of MIDI Info Structure:

Offset	Size	Description (Table 00118)
00h	4 BYTEs	name of the structure ("MIDI")
04h	DWORD	structure length
08h	DWORD	driver software version [00000003h]
0Ch	32 BYTEs	vendor name, etc. (ASCIZ string)
2Ch	32 BYTEs	vendor product name
4Ch	32 BYTEs	vendor chip/hardware description
6Ch	BYTE	installed board number
6Dh	3 BYTEs	unused data
70h	14 BYTEs	the patch library file name [OPL2.BNK 00..]
7Eh	DWORD	feature bits (see #00119)
80h	WORD	user determined preference field
82h	WORD	memory required for driver use
84h	WORD	# of timer tick callbacks per second
86h	WORD	max # of tones (voices, partials)

SeeAlso: #00112,#00120,#00122

Bitfields for MIDI feature bits:

Bit(s)	Description (Table 00119)
0-3	reserved for GM extensions
4	Transmitter/Receiver only
5	Patches preloaded
6	MIDI receive has time stamp
8	MIDI interrupt driven input supported
9	MIDI polled input supported
10	MIDI remote patches supported

Format of MIDI Service structure:

Offset	Size	Description (Table 00120)
00h	4 BYTEs	name of the structure ("MIDS")
04h	DWORD	structure length
08h	16 WORDs	patches loaded table bit field
28h	16 BYTEs	for future expansion
---entry points (details???)---		
38h	DWORD	device check
		11h return available tones
		12h return TRUE/FALSE if patch is understood
		13h set preference
		14h allow/disallow voice stealing
		15h get FIFO sizes
		16h get DMA,IRQ
		17h get IO address
		18h get mem address
		19h get mem free
		80h-.. vendors can add DevChks above 0x80
3Ch	DWORD	global reset
40h	DWORD	MIDI msg
44h	DWORD	poll MIDI
48h	DWORD	preload patch
4Ch	DWORD	unload patch
50h	DWORD	timer tick
54h	DWORD	get last error
		01h unsupported feature/function
		02h unknown patch type (see #00121)
		03h all tones are used
		04h messages are out of sync
		05h an incoming patch was incomplete
		06h an incoming patch couldn't be stored
		07h had to drop an incoming byte
		08h driver is failing a patch download
		80h-.. vendors specific errors
58h	DWORD	Patch Block free callback
5Ch	DWORD	MIDI byte avail. callback

SeeAlso: #00116,#00124

(Table 00121)
Values for MIDI Registered Patch Types:
10h OPL2
11h OPL3

Format of Volume Info Structure:

Offset	Size	Description (Table 00122)
00h	4 BYTEs	name of the structure ("VOLI")
04h	DWORD	structure length (00000092h)
08h	DWORD	driver software version [00000001h]
0Ch	32 BYTEs	vendor name, etc. (ASCIZ string)
2Ch	32 BYTEs	vendor product name
4Ch	32 BYTEs	vendor chip/hardware description
6Ch	BYTE	installed board number (0 for 1st/only board)
6Dh	3 BYTEs	unused data (0)
70h	24 BYTEs	text name of the mixer channel
88h	DWORD	features bits (see #00123)
8Ch	WORD	minimum volume setting
8Eh	WORD	maximum volume setting
90h	WORD	attenuation/gain crossover

SeeAlso: #00112,#00124

Bitfields for Volume feature bits:

Bit(s)	Description (Table 00123)
0	Stereo Volume control available
2	Low Pass Filter is available
3	High Pass Filter is available
4	Parametric Tone Control is available
5	selectable output paths
8	Azimuth Field positioning supported
9	Phi Field positioning supported
10-30	unused???
31	Master Volume device

Format of Volume Services Structure:

Offset	Size	Description (Table 00124)
00h	4 BYTEs	name of the structure ("VOLS")
04h	DWORD	structure length (00000038h)
08h	16 BYTEs	16 bytes for future expansion (0)

---entry points (details???)---

18h	DWORD	device check
		0011h filter range
		0012h filter setting
		0013h filter current
		0014h tone range
		0015h tone setting
		0016h tone current
		0017h path

```
                0018h get IO address
                0080h-.. vendors can add DevChks above 0x80
 1Ch    DWORD          set vol to an absolute setting
                01h User master volume setting
                02h application master volume setting
 20h    DWORD          set 3D volume
 24h    DWORD          tone control
 28h    DWORD          filter control
 2Ch    DWORD          output path
 30h    DWORD          reset channel
 34h    DWORD          get last error
                01h unsupported feature/function
                02h out of range parameter value
                80h+ vendor-specific errors
SeeAlso: #00116,#00120
--------s-104F13BX0003-----------------------
INT 10 - VESA VBE/AI (Audio Interface) - OPEN DEVICE
        AX = 4F13h
        BX = 0003h
        CX = handle
        DX = API set (16/32-bit)
        SI = segment ???
Return: AL = 4Fh if function supported
           AH = status
                00h successful
                   SI:CX -> memory ???
                01h failed
SeeAlso: AX=4F13h/BX=0000h,AX=4F13h/BX=0002h,AX=4F13h/BX=0004h
--------s-104F13BX0004-----------------------
INT 10 - VESA VBE/AI (Audio Interface) - CLOSE DEVICE
        AX = 4F13h
        BX = 0004h
        CX = handle
Return: AL = 4Fh if function supported
           AH = status
                00h successful
                01h failed
SeeAlso: AX=4F13h/BX=0000h,AX=4F13h/BX=0003h,AX=4F13h/BX=0005h
--------s-104F13BX0005-----------------------
INT 10 - VESA VBE/AI (Audio Interface) - UNINSTALL DRIVER
        AX = 4F13h
        BX = 0005h
Return: AL = 4Fh if function supported
           AH = status
                00h successful
                01h failed
SeeAlso: AX=4F13h/BX=0000h,AX=4F13h/BX=0006h
--------s-104F13BX0006-----------------------
```

```
INT 10 - VESA VBE/AI (Audio Interface) - DRIVER CHAIN/UNCHAIN
        AX = 4F13h
        BX = 0006h
Return: AL = 4Fh if function supported
           AH = status
                00h successful
                01h failed
SeeAlso: AX=4F13h/BX=0000h,AX=4F13h/BX=0005h
INT 13 - DISK - GET DRIVE PARAMETERS (PC,XT286,CONV,PS,ESDI,SCSI)
        AH = 08h
        DL = drive (bit 7 set for hard disk)
        ES:DI = 0000h:0000h to guard against BIOS bugs
Return: CF set on error
           AH = status (07h) (see #00234)
        CF clear if successful
           AH = 00h
           AL = 00h on at least some BIOSes
           BL = drive type (AT/PS2 floppies only) (see #00242)
           CH = low eight bits of maximum cylinder number
           CL = maximum sector number (bits 5-0)
                high two bits of maximum cylinder number (bits 7-6)
           DH = maximum head number
           DL = number of drives
           ES:DI -> drive parameter table (floppies only)
Notes:  may return successful even though specified drive is greater than the
          number of attached drives of that type (floppy/hard); check DL to
          ensure validity
        for systems predating the IBM AT, this call is only valid for hard
          disks, as it is implemented by the hard disk BIOS rather than the
          ROM BIOS
        the IBM ROM-BIOS returns the total number of hard disks attached
          to the system regardless of whether DL >= 80h on entry.
        Toshiba laptops with HardRAM return DL=02h when called with DL=80h,
          but fail on DL=81h.  The BIOS data at 40h:75h correctly reports 01h.
        may indicate only two drives present even if more are attached; to
          ensure a correct count, one can use AH=15h to scan through possible
          drives
        Reportedly some Compaq BIOSes with more than one hard disk controller
          return only the number of drives DL attached to the corresponding
          controller as specified by the DL value on entry.  However, on
          Compaq machines with "COMPAQ" signature at F000h:FFEAh,
          MS-DOS/PC DOS IO.SYS/IBMBIO.COM call INT 15/AX=E400h and
          INT 15/AX=E480h to enable Compaq "mode 2" before retrieving the count
          of hard disks installed in the system (DL) from this function.
        the maximum cylinder number reported in CX is usually two less than
          the total cylinder count reported in the fixed disk parameter table
          (see INT 41h,INT 46h) because early hard disks used the last cylinder
          for testing purposes; however, on some Zenith machines, the maximum
```

cylinder number reportedly is three less than the count in the fixed disk parameter table.
for BIOSes which reserve the last cylinder for testing purposes, the cylinder count is automatically decremented
on PS/1s with IBM ROM DOS 4, nonexistent drives return CF clear, BX=CX=0000h, and ES:DI = 0000h:0000h
machines with lost CMOS memory may return invalid data for floppy drives. In this situation CF is cleared, but AX,BX,CX,DX,DH,DI, and ES contain only 0. At least under some circumstances, MS-DOS/PC DOS IO.SYS/IBMBIO.COM just assumes a 360 KB floppy if it sees CH to be zero for a floppy.
the PC-Tools PCFORMAT program requires that AL=00h before it will proceed with the formatting
if this function fails, an alternative way to retrieve the number of floppy drives installed in the system is to call INT 11h.
In fact, the MS-DOS/PC-DOS IO.SYS/IBMBIO.COM attempts to get the number of floppy drives installed from INT 13/AH=08h, when INT 11h AX bit 0 indicates there are no floppy drives installed. In addition to testing the CF flag, it only trusts the result when the number of sectors (CL preset to zero) is non-zero after the call.

BUGS: several different Compaq BIOSes incorrectly report high-numbered drives (such as 90h, B0h, D0h, and F0h) as present, giving them the same geometry as drive 80h; as a workaround, scan through disk numbers, stopping as soon as the number of valid drives encountered equals the value in 0040h:0075h
a bug in Leading Edge 8088 BIOS 3.10 causes the DI,SI,BP,DS, and ES registers to be destroyed
some Toshiba BIOSes (at least before 1995, maybe some laptops??? with 1.44 MB floppies) have a bug where they do not set the ES:DI vector even for floppy drives. Hence these registers should be preset with zero before the call and checked to be non-zero on return before using them. Also it seems these BIOSes can return wrong info in BL and CX, as S/DOS 1.0 can be configured to preset these registers as for an 1.44 MB floppy.
the PS/2 Model 30 fails to reset the bus after INT 13/AH=08h and INT 13/AH=15h. A workaround is to monitor for these functions and perform a transparent INT 13/AH=01h status read afterwards. This will reset the bus. The MS-DOS 6.0 IO.SYS takes care of this by installing a special INT 13h interceptor for this purpose.
AD-DOS may leave interrupts disabled on return from this function.
Some Microsoft software explicitly sets STI after return.

SeeAlso: AH=06h"Adaptec",AH=13h"SyQuest",AH=48h,AH=15h,INT 1E
SeeAlso: INT 41"HARD DISK 0"

(Table 00242)
Values for diskette drive type:

01h	360K
02h	1.2M

```
 03h    720K
 04h    1.44M
 05h    ??? (reportedly an obscure drive type shipped on some IBM machines)
        2.88M on some machines (at least AMI 486 BIOS)
 06h    2.88M
 10h    ATAPI Removable Media Device
--------b-1584-------------------------------
INT 15 - V20-XT-BIOS - JOYSTICK SUPPORT
        AH = 84h
        DX = subfunction
           0000h read joystick switches
                Return: AL bits 7-4 = switch settings
           other: read positions of joysticks as indicated by bits 0-3
                Return: AX = X position of joystick A (if DX bit 0 set)
                        BX = Y position of joystick A (if DX bit 1 set)
                        CX = X position of joystick B (if DX bit 2 set)
                        DX = Y position of joystick B (if DX bit 3 set)
Return: CF set on error
           AH = status (see #00496)
        CF clear if successful
Program: V20-XT-BIOS is a ROM BIOS replacement with extensions by Peter
          Koehlmann / c't magazine
SeeAlso: AH=84h"PS",INT 10/AH=0Eh/CX=ABCDh
--------B-1B--------------------------------
INT 1B C - KEYBOARD - CONTROL-BREAK HANDLER
Desc:   this interrupt is automatically called when INT 09 determines that
          Control-Break has been pressed
Note:   normally points to a short routine in DOS which sets the Ctrl-C flag,
          thus invoking INT 23h the next time DOS checks for Ctrl-C.
SeeAlso: INT 23,MEM 0040h:0071h
--------B-1C--------------------------------
INT 1C - TIME - SYSTEM TIMER TICK
Desc:   this interrupt is automatically called on each clock tick by the INT 08
          handler
Notes:  this is the preferred interrupt to chain when a program needs to be
          invoked regularly
        not available on NEC 9800-series PCs
SeeAlso: INT 08,INT E2"PC Cluster"
--------D-2100------------------------------
INT 21 - DOS 1+ - TERMINATE PROGRAM
        AH = 00h
        CS = PSP segment
Notes:  Microsoft recommends using INT 21/AH=4Ch for DOS 2+
        this function sets the program's return code (ERRORLEVEL) to 00h
        execution continues at the address stored in INT 22 after DOS performs
          whatever cleanup it needs to do (restoring the INT 22,INT 23,INT 24
          vectors from the PSP assumed to be located at offset 0000h in the
          segment indicated by the stack copy of CS, etc.)
```

```
        if the PSP is its own parent, the process's memory is not freed; if
          INT 22 additionally points into the terminating program, the
          process is effectively NOT terminated
        not supported by MS Windows 3.0 DOSX.EXE DOS extender
SeeAlso: AH=26h,AH=31h,AH=4Ch,INT 20,INT 22
--------D-2101-----------------------------
INT 21 - DOS 1+ - READ CHARACTER FROM STANDARD INPUT, WITH ECHO
        AH = 01h
Return: AL = character read
Notes:  ^C/^Break are checked, and INT 23 executed if read
        ^P toggles the DOS-internal echo-to-printer flag
        ^Z is not interpreted, thus not causing an EOF if input is redirected
        character is echoed to standard output
        standard input is always the keyboard and standard output the screen
          under DOS 1.x, but they may be redirected under DOS 2+
SeeAlso: AH=06h,AH=07h,AH=08h,AH=0Ah
--------v-21010F----------------------------
INT 21 - VIRUS - "Susan" - INSTALLATION CHECK
        AX = 010Fh
Return: AX = 7553h ("Su") if resident
SeeAlso: INT 16/AH=DDh"VIRUS",INT 21/AX=0B56h
--------D-2102-----------------------------
INT 21 - DOS 1+ - WRITE CHARACTER TO STANDARD OUTPUT
        AH = 02h
        DL = character to write
Return: AL = last character output (despite the official docs which state
                 nothing is returned) (at least DOS 2.1-7.0)
Notes:  ^C/^Break are checked, and INT 23 executed if pressed
        standard output is always the screen under DOS 1.x, but may be
          redirected under DOS 2+
        the last character output will be the character in DL unless DL=09h
          on entry, in which case AL=20h as tabs are expanded to blanks
        if standard output is redirected to a file, no error checks (write-
          protected, full media, etc.) are performed
SeeAlso: AH=06h,AH=09h
--------D-2103-----------------------------
INT 21 - DOS 1+ - READ CHARACTER FROM STDAUX
        AH = 03h
Return: AL = character read
Notes:  keyboard checked for ^C/^Break, and INT 23 executed if detected
        STDAUX is usually the first serial port
SeeAlso: AH=04h,INT 14/AH=02h,INT E0/CL=03h
--------D-2104-----------------------------
INT 21 - DOS 1+ - WRITE CHARACTER TO STDAUX
        AH = 04h
        DL = character to write
Notes:  keyboard checked for ^C/^Break, and INT 23 executed if detected
        STDAUX is usually the first serial port
```

```
        if STDAUX is busy, this function will wait until it becomes free
SeeAlso: AH=03h,INT 14/AH=01h,INT E0/CL=04h
--------D-2105-----------------------------
INT 21 - DOS 1+ - WRITE CHARACTER TO PRINTER
        AH = 05h
        DL = character to print
Notes:  keyboard checked for ^C/^Break, and INT 23 executed if detected
        STDPRN is usually the first parallel port, but may be redirected under
          DOS 2+
        if the printer is busy, this function will wait
SeeAlso: INT 17/AH=00h
--------D-2131-----------------------------
INT 21 - DOS 2+ - TERMINATE AND STAY RESIDENT
        AH = 31h
        AL = return code
        DX = number of paragraphs to keep resident
Return: never
Notes:  the value in DX only affects the memory block containing the PSP;
          additional memory allocated via AH=48h is not affected
        the minimum number of paragraphs which will remain resident is 11h
          for DOS 2.x and 06h for DOS 3.0+
        most TSRs can save some memory by releasing their environment block
          before terminating (see #01378 at AH=26h,AH=49h)
        any open files remain open, so one should close any files which will
          not be used before going resident; to access a file which is left
          open from the TSR, one must switch PSP segments first (see AH=50h)
SeeAlso: AH=00h,AH=4Ch,AH=4Dh,INT 20,INT 22,INT 27
--------D-2132-----------------------------
INT 21 - DOS 2+ - GET DOS DRIVE PARAMETER BLOCK FOR SPECIFIC DRIVE
        AH = 32h
        DL = drive number (00h = default, 01h = A:, etc)
Return: AL = status
            00h successful
                  DS:BX -> Drive Parameter Block (DPB) (see #01395) for specified
                              drive
            FFh invalid or network drive
Notes:  the OS/2 compatibility box supports the DOS 3.3 version of this call
          except for the DWORD at offset 12h
        this call updates the DPB by reading the disk; the DPB may be accessed
          via the DOS list of lists (see #01627 at AH=52h) if disk access is not
          desirable.
        undocumented prior to the release of DOS 5.0; only the DOS 4.0+
          version of the DPB has been documented, however
        supported by DR DOS 3.41+; DR DOS 3.41-6.0 return the same data as
          MS-DOS 3.31
        IBM ROM-DOS v4.0 also reports invalid/network (AL=FFh) on the ROM drive
SeeAlso: AH=1Fh,AH=52h,AX=7302h
```

```
Format of DOS Drive Parameter Block:
Offset  Size     Description      (Table 01395)
 00h    BYTE     drive number (00h = A:, 01h = B:, etc)
 01h    BYTE     unit number within device driver
 02h    WORD     bytes per sector
 04h    BYTE     highest sector number within a cluster
 05h    BYTE     shift count to convert clusters into sectors
 06h    WORD     number of reserved sectors at beginning of drive
 08h    BYTE     number of FATs
 09h    WORD     number of root directory entries
 0Bh    WORD     number of first sector containing user data
 0Dh    WORD     highest cluster number (number of data clusters + 1)
                 16-bit FAT if greater than 0FF6h, else 12-bit FAT
 0Fh    BYTE     number of sectors per FAT
 10h    WORD     sector number of first directory sector
 12h    DWORD          address of device driver header (see #01646)
 16h    BYTE     media ID byte (see #01356)
 17h    BYTE     00h if disk accessed, FFh if not
 18h    DWORD          pointer to next DPB
---DOS 2.x---
 1Ch    WORD     cluster containing start of current directory, 0000h=root,
                 FFFFh = unknown
 1Eh 64 BYTEs    ASCIZ pathname of current directory for drive
---DOS 3.x---
 1Ch    WORD     cluster at which to start search for free space when writing
 1Eh    WORD     number of free clusters on drive, FFFFh = unknown
---DOS 4.0-6.0---
 0Fh    WORD     number of sectors per FAT
 11h    WORD     sector number of first directory sector
 13h    DWORD          address of device driver header (see #01646)
 17h    BYTE     media ID byte (see #01356)
 18h    BYTE     00h if disk accessed, FFh if not
 19h    DWORD          pointer to next DPB
 1Dh    WORD     cluster at which to start search for free space when writing,
                 usually the last cluster allocated
 1Fh    WORD     number of free clusters on drive, FFFFh = unknown
SeeAlso: #01357,#01663,#01787 at AX=7302h,#04039 at INT E0/CL=71h
--------D-213305----------------------------
INT 21 - DOS 4.0+ - GET BOOT DRIVE
        AX = 3305h
Return: DL = boot drive (1=A:,...)
Notes:  This function does not use any of the DOS-internal stacks and may
          thus be called at any time.  It is directly dispatched from
          the INT 21h entry point with interrupts disabled.
        NEC 9800-series PCs always call the boot drive A: and assign the other
         drive letters sequentially to the other drives in the system
        this call is supported by OS/2 Warp 3.0, but not earlier versions of
         OS/2; it is also supported by Novell DOS 7
```

```
--------D-215D0B----------------------------
INT 21 OU - DOS 4.x only - internal - GET DOS SWAPPABLE DATA AREAS
        AX = 5D0Bh
Return: CF set on error
           AX = error code (see #01680)
        CF clear if successful
           DS:SI -> swappable data area list (see #01689)
Notes:  copying and restoring the swappable data areas allows DOS to be
          reentered unless it is in a critical section delimited by calls to
          INT 2A/AH=80h and INT 2A/AH=81h,82h
        SHARE and other DOS utilities consult the byte at offset 04h in the
          DOS data segment (see INT 2F/AX=1203h) to determine the SDA format
          in use: 00h = DOS 3.x, 01h = DOS 4.0-6.0, other = error.
        DOS 5+ use the SDA format listed below, but revert back to the DOS 3.x
          call for finding the SDA (see #01687); Novell DOS 7 does not support
          this function, either.
SeeAlso: AX=5D06h,INT 2A/AH=80h,INT 2A/AH=81h,INT 2A/AH=82h,INT 2F/AX=1203h

Format of DOS 4.x swappable data area list:
Offset  Size     Description      (Table 01689)
 00h    WORD     count of data areas
 02h  N BYTEs    "count" copies of data area record
                 Offset  Size     Description
                  00h    DWORD          address
                  04h    WORD     length and type
                                  bit 15 set if swap always, clear if swap in DOS
                                  bits 14-0: length in bytes
SeeAlso: #01690

Format of DOS 4.0-6.0 swappable data area:
Offset  Size     Description      (Table 01690)
 -34    BYTE     printer echo flag (00h off, FFh active)
 -31    BYTE     current switch character (ignored by DOS 5+)
 -30    BYTE     current memory allocation strategy (see AH=58h)
 -28    BYTE     incremented on each INT 21/AX=5E01h call
 -27 16 BYTEs    machine name set by INT 21/AX=5E01h
 -11  5 WORDs    zero-terminated list of offsets which need to be patched to
                   enable critical-section calls (see INT 2A/AH=80h)
                 (all offsets are 0D0Ch, but this list is still present for
                   DOS 3.x compatibility)
 -1     BYTE     unused padding
Note:   the above data is not actually part of the SDA, and is much more likely
          to change between DOS versions/OEMs than data in the SDA itself
---start of actual SDA---
 00h    BYTE     critical error flag ("ErrorMode")
 01h    BYTE     InDOS flag (count of active INT 21 calls)
 02h    BYTE     drive on which current critical error occurred or FFh
                 (DR DOS 3.41/5.0 set this to 00h when no critical error)
```

```
 03h    BYTE    locus of last error
 04h    WORD    extended error code of last error
 06h    BYTE    suggested action for last error
 07h    BYTE    class of last error
 08h    DWORD           ES:DI pointer for last error
 0Ch    DWORD           current DTA (Disk Transfer Address)
                 note: may point into SDA during the DOS EXEC function
                   (see AH=4Bh), so programs which swap the SDA must be
                   prepared to move the DTA to a private buffer if they
                   might be invoked during an EXEC
 10h    WORD    current PSP
 12h    WORD    stores SP across an INT 23
 14h    WORD    return code from last process termination (zerod after reading
                   with AH=4Dh)
 16h    BYTE    current drive
 17h    BYTE    extended break flag
 18h    BYTE    flag: code page switching
 19h    BYTE    flag: copy of previous byte in case of INT 24 Abort
---remainder need only be swapped if in DOS---
 1Ah    WORD    value of AX on call to INT 21
                 Note:   does not contain correct value on functions 00h-0Ch,
                           50h, 51h, 59h, or 62h
 1Ch    WORD    PSP segment for sharing/network (0000h = local)
 1Eh    WORD    network machine number for sharing/network (0000h = local)
 20h    WORD    first usable memory block found when allocating memory
 22h    WORD    best usable memory block found when allocating memory
 24h    WORD    last usable memory block found when allocating memory
 26h    WORD    memory size in paragraphs (used only during initialization)
 28h    WORD    last entry checked during directory search
 2Ah    BYTE    flag: nonzero if INT 24 Fail
 2Bh    BYTE    flags: allowable INT 24 responses (passed to INT 24 in AH)
 2Ch    BYTE    flag: do not set directory if nonzero
 2Dh    BYTE    flag: program aborted by ^C
 2Eh    BYTE    flag: allow embedded blanks in FCB
                 may also allow use of "*" wildcard in FCBs
 2Fh    BYTE    padding (unused)
 30h    BYTE    day of month
 31h    BYTE    month
 32h    WORD    year - 1980
 34h    WORD    number of days since 01jan1980
 36h    BYTE    day of week (0 = Sunday)
 37h    BYTE    flag: console swapped during read from device
 38h    BYTE    flag: safe to call INT 28 if nonzero
 39h    BYTE    flag: abort currently in progress, turn INT 24 Abort into Fail
 3Ah 30 BYTEs   device driver request header (see #02597 at INT 2F/AX=0802h) for
                   device calls
 58h    DWORD           pointer to device driver entry point (used in calling driver)
 5Ch 22 BYTEs   device driver request header for I/O calls
```

72h	14 BYTEs	device driver request header for disk status check (also includes following eight bytes for some calls)
80h	DWORD	pointer to device I/O buffer
84h	WORD	part of request header at 72h
86h	WORD	part of request header at 72h (0)
88h	BYTE	type of PSP copy (00h=simple for INT 21/AH=26h, FFh=make child)
89h	DWORD	start offset of file region to lock/unlock
8Dh	DWORD	length of file region to lock/unlock
91h	BYTE	padding (unused)
92h	3 BYTEs	24-bit user number (see AH=30h)
95h	BYTE	OEM number (see #01394 at AH=30h)
96h	6 BYTEs	CLOCK$ transfer record (see #01688 at AX=5D06h)
9Ch	BYTE	device I/O buffer for single-byte I/O functions
9Dh	BYTE	padding
9Eh	128 BYTEs	buffer for filename
11Eh	128 BYTEs	buffer for filename (rename destination name)
19Eh	21 BYTEs	findfirst/findnext search data block (see #01626 at AH=4Eh)
1B3h	32 BYTEs	directory entry for found file (see #01394 at AH=11h)
1D3h	88 BYTEs	copy of current directory structure for drive being accessed
22Bh	11 BYTEs	FCB-format filename for device name comparison
236h	BYTE	terminating NUL for above filename
237h	11 BYTEs	wildcard destination specification for rename (FCB format)
242h	BYTE	terminating NUL for above filespec
243h	BYTE	padding???
244h	WORD	destination starting sector (cluster???)
246h	5 BYTEs	extra space to allow a directory entry to be stored starting at offset 22Bh
24Bh	BYTE	extended FCB file attributes
24Ch	BYTE	type of FCB (00h regular, FFh extended)
24Dh	BYTE	directory search attributes
24Eh	BYTE	file open/access mode
24Fh	BYTE	flag: nonzero if file was deleted
250h	BYTE	flag: device name found on rename, or file not found
251h	BYTE	flag: splice file name and directory name together
252h	BYTE	flag indicating how DOS function was invoked (00h = direct INT 20/INT 21, FFh = server call AX=5D00h)
253h	BYTE	sector position within cluster
254h	BYTE	flag: translating sector/cluster
255h	BYTE	flag: 00h if read, 01h if write
256h	BYTE	current working drive number
257h	BYTE	cluster factor
258h	BYTE	"sda_CLUSSPLIT" flag: cluster split between two FAT sectors
259h	BYTE	line edit (AH=0Ah) insert mode flag (nonzero = on)
25Ah	BYTE	canonicalized filename referred to existing file/dir if FFh
25Bh	BYTE	volume ID flag
25Ch	BYTE	type of process termination (00h-03h) (see AH=4Dh)
25Dh	BYTE	unused (padding for alignment)
25Eh	BYTE	file create flag (00h = no, search only)

```
25Fh   BYTE    value for deleted file's first byte: 00h to delete all, else E5
260h   DWORD          pointer to Drive Parameter Block for critical error invocation
264h   DWORD          pointer to stack frame containing user registers on INT 21
268h   WORD  stores SP across INT 24
26Ah   DWORD          pointer to DOS Drive Parameter Block for ???
26Eh   WORD  segment of disk buffer
270h   DWORD          saving partial cluster number
274h   WORD  "sda_PREREAD" 00h if preread, 01h if optional
276h   WORD  temporary used in allocating disk space
278h   BYTE    Media ID byte returned by AH=1Bh,1Ch
279h   BYTE    unused
27Ah   DWORD          pointer to device header if filename is character device
27Eh   DWORD          pointer to current SFT
282h   DWORD          pointer to current directory structure for drive being accessed
286h   DWORD          pointer to caller's FCB
28Ah   WORD  SFT index to which file being opened will refer
28Ch   WORD  temporary storage for file handle
28Eh   DWORD          pointer to JFT entry (for file being opened) in process handle
                 table (see #01378 at AH=26h)
292h   WORD  "sda_WFP_START" offset in DOS DS of first filename argument
294h   WORD  "sda_REN_WFP" offset in DOS DS of second filename argument
296h   WORD  offset of last component in pathname or FFFFh
298h   WORD  offset of transfer address to add
29Ah   WORD  last relative cluster within file being accessed
29Ch   WORD  temp: absolute cluster number being accessed
29Eh   DWORD          directory sector number
2A2h   WORD  directory cluster number
2A4h   DWORD          current relative sector number within file
2A8h   DWORD          current sector number (number of previously written sectors)
2ACh   WORD  current byte offset within sector
2AEh   DWORD          current offset in file
2B2h   WORD  number of bytes in first sector
2B4h   WORD  bytes in partial last sector
2B6h   WORD  number of whole sectors
2B8h   WORD  free file cluster entry
2BAh   WORD  last file cluster entry
2BCh   WORD  next file cluster number
2BEh   DWORD          number of bytes appended to file
2C2h   DWORD          pointer to current work disk buffer
2C6h   DWORD          pointer to working SFT
2CAh   WORD  used by INT 21 dispatcher to store caller's BX
2CCh   WORD  used by INT 21 dispatcher to store caller's DS
2CEh   WORD  temporary storage while saving/restoring caller's registers
2D0h   DWORD          pointer to prev call frame (offset 264h) if INT 21 reentered
               also switched to for duration of INT 24
2D4h   WORD  open mode/action for INT 21/AX=6C00h
2D6h   BYTE    extended open conditional flag
               set to 00h by INT 21h dispatcher, 02h when a read is
```

```
                    performed, and 01h or 03h by INT 21/AX=6C00h
2D7h    WORD  extended open I/O mode
2D9h    DWORD         stored ES:DI for AX=6C00h
2DDh    WORD  extended file open action code (see #01770 at AX=6C00h)
2DFh    WORD  extended file open attributes (see #01769 at AX=6C00h)
2E1h    WORD  extended file open file mode (see AX=6C00h)
2E3h    DWORD         pointer to filename to open (see AX=6C00h)
2E7h    WORD  high word of 32-bit sector number, or temp data buffer size
                  from disk buffer
2E9h    WORD  "sda_OffsetMagicPatch"
2EBh    BYTE  disk full on >32M partition when set to 01h
2ECh    WORD  stores DS during call to [List-of-Lists + 37h]
2EEh    WORD  temporary storage (various uses)
2F0h    BYTE  storage for drive error
2F1h    WORD  DOS 3.4 (European MS-DOS 4.00) bit flags
2F3h    DWORD         pointer to user-supplied filename
2F7h    DWORD         pointer to user-supplied rename destination filename
2FBh    WORD  stores SS during call to [List-of-Lists + 37h] and INT 25,26
2FDh    WORD  stores SP during call to [List-of-Lists + 37h] and INT 25,26
2FFh    BYTE  flag, nonzero if stack switched in calling [List-of-Lists+37h]
300h 21 BYTEs FindFirst search data for source file(s) of a rename operation
              (see #01626 at AH=4Eh)
315h 32 BYTEs directory entry for file being renamed (see #01352 at AH=11h)
335h 331 BYTEs        critical error stack
480h 384 BYTEs        disk stack (functions greater than 0Ch, INT 25,INT 26)
600h 384 BYTEs        character I/O stack (functions 01h through 0Ch)
780h    BYTE  device driver lookahead flag (usually printer)
              (see AH=64h"DOS 3.2+")
781h    BYTE  volume change flag
782h    BYTE  flag: virtual file open
783h    BYTE  fastseek drive
784h    WORD  fastseek first cluster number
786h    WORD  fastseek logical cluster number
788h    WORD  fastseek returned logical cluster number
78Ah    WORD  temporary location of DOS@SYSINIT
---MSDOS 7.1+ (FAT32)---
78Ch 47 BYTEs ???
7BBh    BYTE  flag: absolute disk read/write type
              00h = INT 25/INT 26
              01h = INT 21/AX=7305h
7BCh    WORD  high word of directory cluster number at offset 2A2h
7BEh    WORD  high word of cluster number at offset 29Ch
7C0h    WORD  high word of next file cluster number at offset 2BCh
7C2h    WORD  high word of last relative cluster number at offset 29Ah
7C4h    WORD  high word of temp at offset 276h
7C6h    WORD  high word of offset 244h
7C8h    WORD  high word of EBX
7CAh    WORD  high word of EDX used by "PACK"
```

```
7CCh    WORD  high word of EDI used by "UNPACK"
7CEh    WORD  high word of EBX used by "SETDIRSRCH"
7D0h    WORD  high word of ECX used by "FREECLUSTER"
7D2h    WORD  high word of EDI used by "GETEOF"
7D4h  3 WORDs          ???
Note:   the only fields which remain valid BETWEEN calls to INT 21h are those
          in the initial "swap-always" portion of the SDA
SeeAlso: #01687,#01689
--------D-215E00----------------------------
INT 21 - DOS 3.1+ network - GET MACHINE NAME
        AX = 5E00h
        DS:DX -> 16-byte buffer for ASCII machine name
Return: CF clear if successful
           CH = validity
                00h name invalid
                nonzero valid
                   CL = NetBIOS number for machine name
                   DS:DX buffer filled with blank-paded name
        CF set on error
           AX = error code (01h) (see #01680 at AH=59h)
Note:   supported by OS/2 v1.3+ compatibility box, PC-NFS
SeeAlso: AX=5E01h

--------D-2171------------------------------
INT 21 - Windows95 - LONG FILENAME FUNCTIONS
        AH = 71h
        AL = function
           0Dh reset drive (see AX=710Dh)
           39h create directory (see AX=7139h)
           3Ah remove directory (see AX=713Ah)
           3Bh set current directory (see AX=713Bh)
           41h delete file (see AX=7141h)
           43h get/set file attributes (see AX=7143h)
           47h get current directory (see AX=7147h)
           4Eh find first file (see AX=714Eh)
           4Fh find next file (see AX=714Fh)
           56h move (rename) file (see AX=7156h)
           60h truename (see AX=7160h/CL=00h,AX=7160h/CL=02h)
           6Ch create/open file (see AX=716Ch)
           A0h get volume information (see AX=71A0h)
           A1h terminate FindFirst/FindNext (see AX=71A1h)
           A6h get file information (see AX=71A6h)
           A7h time conversion (see AX=71A7h/BL=00h,AX=71A7h/BL=01h)
           A8h generate short filename (see AX=71A8h)
           A9h server create/open file (see AX=71A9h)
           AAh create/terminate SUBST (see AX=71AAh/BH=00h,AX=71AAh/BH=02h)
Return: CF set on error
           AX = error code (see #01680)
```

```
                7100h if function not supported
        CF clear if successful
           other registers as for corresponding "old" DOS function
Notes:  if error 7100h is returned, the old-style function should be called
        AX=714Eh returns a "search handle" which must be passed to AX=714Fh;
          when the search is complete, AX=71A1h must be called to terminate
          the search
        for compatibility with DOS versions prior to v7.00, the carry flag
          should be set on call to ensure that it is set on exit
        Caldera's DPMS-enabled LONGNAME.EXE BETA 1 extension for DR-DOS 7
          supports the following sub-set of LFN functions: 39h, 3Ah, 3Bh, 41h,
          43h (BL = 0, 1 only), 47h, 4Eh, 4Fh, 56h, 60h (CL = 0, 1, 2), 6Ch,
          A0h, A1h, A8h. BETA 2 fixes LFN directory entry checksums, which
          were causing wrong LFNs to be attached to a file. The 8.3 short
          names for filenames with exactly 8 chars are no longer abbreviated
          (e.g. LONGNAME.TXT -> LONGNAME.TXT, not LONGNA~1.TXT). BETA 3 has
          A7h (BL=0, 1) functions added, and 4Eh/4Fh can return file times
          in both DOS and 64 bit formats, BETA 4 has support added for
          Caldera's DRFAT32 redirector extension (see INT 2F/AX=15xxh).
        Caldera's DR-OpenDOS 7.02+ COMMAND.COM utilizes the LFN API as soon
          as it detects it (mind, that LONGNAME.EXE can be dynamically loaded
          and unloaded at runtime). This COMMAND.COM shell also works under
          MS-DOS/PC DOS and in DOS boxes of Windows9x, NT, 2000, and OS/2.
        For 4DOS 6.02+ to work with 3rd party LFN providers, the Win95LFN=Yes
          directive should be inserted into the 4DOS.INI file.
        Mike Podanoffsky's RxDOS 7.2 provides most of this API natively,
          including functions 39h, 3Ah, 3Bh, 41h, 43h (BL = ???), 47h, 4Bh,
          4Eh, 4Fh, 56h, 60h (CL = 0, 1, 2, no CH), 6Ch, A0h, A1h and A7h.
          However, not all sub-functions seem to be supported yet.
SeeAlso: AH=39h,AH=3Ah,AH=3Bh,AH=41h,AX=4300h,AX=4301h,AX=4304h,AX=4306h
SeeAlso: AX=4307h,AH=47h,AH=4Eh,AH=4Fh,AH=56h,AH=6Ch,AX=714Eh,AX=714Fh
--------N-21E1--SF04-------------------------
INT 21 O - Novell NetWare - MESSAGE SERVICES - SEND PERSONAL MESSAGE
        AH = E1h subfn 04h
        DS:SI -> request buffer (see #01826)
        ES:DI -> reply buffer (see #01827)
Return: AL = status
           00h successful
           FEh I/O error or out of dynamic workspace
Notes:  this function is supported by NetWare 4.0+ and Advanced NetWare 1.0-2.x
        message pipes use CPU time on the file server; IPX, SPX, or NetBIOS
          connections should be used for peer-to-peer communications as these
          protocols do not use file server time
SeeAlso: AH=E1h/SF=00h,AH=E1h/SF=05h,AH=E1h/SF=06h,AH=E1h/SF=08h

Format of NetWare "Send Personal Message" request buffer:
Offset  Size    Description     (Table 01826)
 00h    WORD    length of following data (max E5h)
```

```
 02h     BYTE    04h (subfunction "Send Personal Message")
 03h     BYTE    number of connections (01h-64h)
 04h  N BYTEs    list of connections to receive broadcast message
         BYTE    length of message (01h-7Eh)
     N BYTEs     message (no control characters or characters > 7Eh)
SeeAlso: #01827

Format of NetWare "Send Personal Message" reply buffer:
Offset  Size     Description      (Table 01827)
 00h     WORD    (call) size of following results buffer (max 65h)
 02h     BYTE    number of connections
 03h  N BYTEs    list of per-connection results
                 00h successful
                 FCh message rejected because queue is full (contains 6 msgs)
                 FDh incomplete pipe
                 FFh failed
SeeAlso: #01826
--------N-21E1--SF05------------------------
INT 21 O - Novell NetWare - MESSAGE SERVICES - GET PERSONAL MESSAGE
         AH = E1h subfn 05h
         DS:SI -> request buffer (see #01828)
         ES:DI -> reply buffer (see #01829)
Return: AL = status
            00h successful
            FEh out of dynamic workspace
Desc:   return the oldest message in the default file server's message queue
          for the calling workstation
Note:   this function is supported by NetWare 4.0+ and Advanced NetWare 1.0-2.x
SeeAlso: AH=E1h/SF=01h,AH=E1h/SF=04h,AH=E1h/SF=06h,AH=E1h/SF=08h

Format of NetWare "Get Personal Message" request buffer:
Offset  Size     Description      (Table 01828)
 00h     WORD    0001h (length of following data)
 02h     BYTE    05h (subfunction "Get Personal Message")
SeeAlso: #01829

Format of NetWare "Get Personal Message" reply buffer:
Offset  Size     Description      (Table 01829)
 00h     WORD    (call) size of following results buffer (max 80h)
 02h     BYTE    connection number of sending station
 03h     BYTE    length of message (00h-7Eh)
                 00h if no personal messages pending
 04h  N BYTEs    message (no control characters or characters > 7Eh)
SeeAlso: #01828
--------D-23--------------------------------
INT 23 - DOS 1+ - CONTROL-C/CONTROL-BREAK HANDLER
---DOS 1.x---
Return: AH = 00h abort program
```