```
Interrupt List, part 7 of 18
Copyright (c) 1989-1999,2000 Ralf Brown
--------D-214400-----------------------------
INT 21 - DOS 2+ - IOCTL - GET DEVICE INFORMATION
  AX = 4400h
  BX = handle
Return: CF clear if successful
      DX = device information word (see #01423)
      AX destroyed
  CF set on error
      AX = error code (01h,05h,06h) (see #01680 at AH=59h/BX=0000h)
Notes:  value in DH corresponds to high byte of device driver's attribute word
    if handle refers to a character device
  Novell NetWare reportedly does not return a drive number in bits 5-0
    for a disk file
  this function was not supported by Digital Research's DOS Plus
SeeAlso: AX=4401h,INT 2F/AX=122Bh

Bitfields for device information word:
Bit(s)  Description (Table 01423)
 character device
  14  device driver can process IOCTL requests (see AX=4402h"DOS 2+")
  13  output until busy supported
  11  driver supports OPEN/CLOSE calls
   8  ??? (set by MS-DOS 6.2x KEYB)
   7  set (indicates device)
   6  EOF on input
   5  raw (binary) mode
   4  device is special (uses INT 29)
   3  clock device
   2  NUL device
   1  standard output
   0  standard input
 disk file
  15  file is remote (DOS 3.0+)
  14  don't set file date/time on closing (DOS 3.0+)
  11  media not removable
   8  (DOS 4 only) generate INT 24 if no disk space on write or read past
    end of file
   7  clear (indicates file)
   6  file has not been written
```

```
  5-0 drive number (0 = A:)
SeeAlso: INT 29
--------D-214401-----------------------------
INT 21 - DOS 2+ - IOCTL - SET DEVICE INFORMATION
  AX = 4401h
  BX = handle (must refer to character device)
  DX = device information word (see #01423)
       (DH must be zero for DOS version prior to 6.x)
Return: CF clear if successful
  CF set on error
       AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Note: this function was not supported by Digital Research's DOS Plus
SeeAlso: AX=4400h,INT 2F/AX=122Bh
--------D-214402-----------------------------
INT 21 - DOS 2+ - IOCTL - READ FROM CHARACTER DEVICE CONTROL CHANNEL
  AX = 4402h
  BX = file handle referencing character device
  CX = number of bytes to read
  DS:DX -> buffer
Return: CF clear if successful
       AX = number of bytes actually read
  CF set on error
       AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Notes:  format of data is driver-specific
     (also see separate entries below for some specific cases)
  this function was not supported by Digital Research's DOS Plus
SeeAlso: AX=4400h,AX=4403h"DOS",AX=4404h"DOS",INT 2F/AX=122Bh
--------N-214402-----------------------------
INT 21 - Network Driver Interface Specification (NDIS) 2.0.1 - PROTOCOL MANAGER
  AX = 4402h
  BX = file handle for device "PROTMAN$"
  CX = 000Eh (size of request block)
  DS:DX -> request block (see #01424,#01425,#01426,#01427,#01428,#01432,#01433)
Return: CF clear if successful
       AX = number of bytes actually read
  CF set on error
       AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
SeeAlso: AX=4402h"FTPSOFT"

Format of NDIS request block for GetProtocolManagerInfo:
Offset  Size  Description (Table 01424)
```

```
 00h  WORD  01h
 02h  WORD  returned status (see #01434)
 04h  DWORD returned pointer to structure representing parsed user config
 08h  DWORD unused
 0Ch  WORD  returned BCD version of NDIS on which Protocol Manager is based
SeeAlso: #01425,#01426,#01427,#01428,#01429,#01430,#01431,#01432,#01433

Format of NDIS request block for RegisterModule:
Offset  Size  Description (Table 01425)
 00h  WORD  02h
 02h  WORD  returned status (see #01434)
 04h  DWORD pointer to module's common characteristics table (see #01435)
 08h  DWORD pointer to list of modules to which the module is to be bound
 0Ch  WORD  unused
SeeAlso: #01424,#01426,#01427,#01428,#01429,#01430,#01431,#01432,#01433

Format of NDIS request block for BindAndStart:
Offset  Size  Description (Table 01426)
 00h  WORD  03h
 02h  WORD  returned status (see #01434)
 04h  DWORD caller's virtual address in FailingModules structure
 08h  DWORD unused
 0Ch  WORD  unused
SeeAlso: #01424,#01425,#01427,#01428,#01429,#01430,#01431,#01432,#01433

Format of NDIS request block for GetProtocolManagerLinkage:
Offset  Size  Description (Table 01427)
 00h  WORD  04h
 02h  WORD  returned status (see #01434)
 04h  DWORD returned dispatch point
 08h  DWORD unused
 0Ch  WORD  returned protocol manager DS
Note: the dispatch point may be called as follows instead of using this IOCTL
  STACK: WORD  protocol manager DS
         DWORD pointer to request block
  Return: AX = returned status
    STACK popped
SeeAlso: #01424,#01425,#01426,#01428,#01429,#01430,#01431,#01432,#01433

Format of NDIS request block for GetProtocolIniPath:
Offset  Size  Description (Table 01428)
```

```
 00h  WORD  05h
 02h  WORD  returned status (see #01434)
 04h  DWORD pointer to a buffer for the ASCIZ pathname of PROTOCOL.INI
 08h  DWORD unused
 0Ch  WORD  buffer length
SeeAlso: #01424,#01425,#01426,#01427,#01429,#01430,#01431,#01432,#01433

Format of NDIS request block for RegisterProtocolManagerInfo:
Offset  Size  Description (Table 01429)
 00h  WORD  06h
 02h  WORD  returned status (see #01434)
 04h  DWORD pointer to structure containing parsed user config file
 08h  DWORD unused
 0Ch  WORD  length of structure
SeeAlso: #01424,#01425,#01426,#01427,#01428,#01430,#01431,#01432,#01433

Format of NDIS request block for InitAndRegister:
Offset  Size  Description (Table 01430)
 00h  WORD  07h
 02h  WORD  returned status (see #01434)
 04h  DWORD unused
 08h  DWORD poitner to ASCIZ name of the module to be prebind initialized
 0Ch  WORD  unused
SeeAlso: #01424,#01425,#01426,#01427,#01428,#01429,#01431,#01432,#01433

Format of NDIS request block for UnbindAndStop:
Offset  Size  Description (Table 01431)
 00h  WORD  08h
 02h  WORD  returned status (see #01434)
 04h  DWORD failing modules as for BindAndStart
 08h  DWORD if not 0000h:0000h, pointer to ASCIZ name of module to unbind
    if 0000h:0000h, terminate a set of previously dynamically
      bound protocol modules
 0Ch  WORD  unused
SeeAlso: #01424,#01425,#01426,#01427,#01428,#01429,#01430,#01432,#01433

Format of NDIS request block for BindStatus:
Offset  Size  Description (Table 01432)
 00h  WORD  09h
 02h  WORD  returned status (see #01434)
 04h  DWORD must be 0000h:0000h
```

```
    on return, points to root tree
 08h  DWORD 0000h:0000h
 0Ch  WORD  unused under DOS
SeeAlso: #01424,#01425,#01426,#01427,#01428,#01429,#01430,#01431,#01433

Format of NDIS request block for RegisterStatus:
Offset  Size  Description (Table 01433)
 00h  WORD  0Ah
 02h  WORD  returned status (0000h, 0008h, 002Ch) (see #01434)
 04h  DWORD 0000h:0000h
 08h  DWORD pointer to 16-byte ASCIZ module name
 0Ch  WORD  0000h
Note: not supported by the 10NET v5.0 PROTMAN$ driver
SeeAlso: #01424,#01425,#01426,#01427,#01428,#01429,#01430,#01431,#01432

(Table 01434)
Values for NDIS status code:
 0000h  success
 0001h  wait for release--protocol has retained control of the data buffer
 0002h  request queued
 0003h  frame not recognized
 0004h  frame rejected
 0005h  frame should be forwarded
 0006h  out of resource
 0007h  invalid parameter
 0008h  invalid function
 0009h  not supported
 000Ah  hardware error
 000Bh  transmit error
 000Ch  unrecognized destination
 000Dh  buffer too small
 0020h  already started
 0021h  binding incomplete
 0022h  driver not initialized
 0023h  hardware not found
 0024h  hardware failure
 0025h  configuration failure
 0026h  interrupt conflict
 0027h  MAC incompatible
 0028h  initialization failed
 0029h  no binding
```

```
 002Ah  network may be disconnected
 002Bh  incompatible OS version
 002Ch  already registered
 002Dh  path not found
 002Eh  insufficient memory
 002Fh  info not found
 00FFh  general failure
 F000h-FFFFh reserved for vendor-specific codes, treated as general failure

Format of NDIS common characteristics table:
Offset  Size  Description (Table 01435)
 00h  WORD  size of table in bytes
 02h  BYTE  NDIS major version
 03h  BYTE  NDIS minor version
 04h  WORD  reserved
 06h  BYTE  module major version
 07h  BYTE  module minor version
 08h  DWORD module function flag bits
    bit 0: binding at upper boundary supported
    bit 1: binding at lower boundary supported
    bit 2: dynamically bound
    bits 3-31 reserved, must be 0
 0Ch 16 BYTEs ASCIZ module name
 1Ch  BYTE  upper boundary protocol level (see #01436)
 1Dh  BYTE  upper boundary interface type
    for MACs: 1 = MAC
    for data links and transports: to be defined
    for session: 1 = NCB
    any level: 0 = private (ISV-defined)
 1Eh  BYTE  lower boundary protocol level (see #01436)
 1Fh  BYTE  lower boundary interface type
    same as offset 1Dh
 20h  WORD  module ID filled in by protocol manager
 22h  WORD  module DS
 24h  DWORD system request entry point
 28h  DWORD pointer to service-specific characteristics (see #01438,#01440)
    0000h:0000h if none
 2Ch  DWORD pointer to service-specific status, or 0000h:0000h if none
      (see #01441)
 30h  DWORD pointer to upper dispatch table (see #01437)
    0000h:0000h if none
```

```
 34h  DWORD pointer to lower dispatch table (see #01437)
    0000h:0000h if none
 38h  2 DWORDs  reserved, must be 0
Note: for compatibility with NDIS 1.x.x, a major version of 00h is
    interpreted as 01h

(Table 01436)
Values for NDIS boundary protocol level:
 00h  physical
 01h  Media Access Control
 02h  Data link
 03h  network
 04h  transport
 05h  session
 FFh  not specified

Format of NDIS dispatch table:
Offset  Size  Description (Table 01437)
 00h  DWORD -> common characteristics table (see #01435)
 04h  4 BYTEs ???
 08h  DWORD -> ??? function (called with 12 bytes of stack arguments)
 0Ch  DWORD -> ??? function (called with 10 bytes of stack arguments)
 10h  DWORD -> ??? function (called with 16 bytes of stack arguments)
 14h  DWORD -> ??? function (called with 4 bytes of stack arguments)
 18h  DWORD -> ??? function (called with 18 bytes of stack arguments)
 1Ch  DWORD -> ??? function (called with 12 bytes of stack arguments)

Format of MAC Service-Specific Characteristics Table:
Offset  Size  Description (Table 01438)
 00h  WORD  length of table in bytes
 02h 16 BYTEs ASCIZ MAC type name, "802.3", "802.4", "802.5", "802.6", "DIX",
      "DIX+802.3", "APPLETALK", "ARCNET", "FDDI", "SDLC", "BSC",
      "HDLC", or "ISDN"
 12h  WORD  length of station addresses in bytes
 14h 16 BYTEs permanent station address
 24h 16 BYTEs current station address
 34h  DWORD current functional adapter address (00000000h if none)
 38h  DWORD pointer to multicast address list
 3Ch  DWORD link speed in bits/sec
 40h  DWORD service flags (see #01439)
 44h  WORD  maximum frame size which may be both sent and received
```

```
 46h  DWORD total transmit buffer capacity in bytes
 4Ah  WORD  transmit buffer allocation block size in bytes
 4Ch  DWORD total receive buffer capacity in bytes
 50h  WORD  receive buffer allocation block size in bytes
 52h  3 BYTEs IEEE vendor code
 55h  BYTE  vendor adapter code
 56h  DWORD pointer to ASCIZ vendor adapter description
 5Ah  WORD  IRQ used by adapter
 5Ch  WORD  transmit queue depth
 5Eh  WORD  maximum supported number of data blocks in buffer descriptors
 60h  N BYTEs vendor-specific info
SeeAlso: #01440

Bitfields for service flags:
Bit(s)  Description (Table 01439)
 0  supports broadcast
 1  supports multicast
 2  supports functional/group addressing
 3  supports promiscuous mode
 4  station address software settable
 5  statistics always current
 6  supports InitiateDiagnostics
 7  supports loopback
 8  MAC does primarily ReceiveChain indications instead of ReceiveLookahead
  indications
 9  supports IBM source routing
 10 supports MAC reset
 11 supports Open/Close adapter
 12 supports interrupt request
 13 supports source routing bridge
 14 supports GDT virtual addresses (OS/2 version)
 15 multiple TransferDatas allowed durign a single indication
 16 MAC normally sets FrameSize = 0 in ReceiveLookahead
 17-31  reserved, must be 0

Format of NetBIOS Service-Specific Characteristics Table:
Offset  Size  Description (Table 01440)
 00h  WORD  length of table in bytes
 02h 16 BYTEs ASCIZ type name of NetBIOS module
 12h  WORD  NetBIOS module code
 14h  N BYTEs vendor-specific info
```

SeeAlso: #01438

Format of MAC Service-Specific Status Table:
Offset Size Description (Table 01441)
 00h WORD length of table in bytes
 02h DWORD seconds since 0:00 1/1/70 when diagnostics last run
    (FFFFFFFFh = never)
 06h DWORD MAC status bits (see #01442)
 0Ah WORD current packet filter flags (see #01443)
 0Ch DWORD pointer to media-specific status table or 0000h:0000h
 10h DWORD seconds past 0:00 1/1/70 of last ClearStatistics
 14h DWORD total frames received (FFFFFFFFh = not counted)
 18h DWORD frames with CRC error (FFFFFFFFh = not counted)
 1Ch DWORD total bytes received (FFFFFFFFh = not counted)
 20h DWORD frames discarded--no buffer space (FFFFFFFFh = not counted)
 24h DWORD multicast frames received (FFFFFFFFh = not counted)
 28h DWORD broadcast frames received (FFFFFFFFh = not counted)
 2Ch DWORD frames with errors (FFFFFFFFh = not counted)
 30h DWORD overly large frames (FFFFFFFFh = not counted)
 34h DWORD frames less than minimum size (FFFFFFFFh = not counted)
 38h DWORD multicast bytes received (FFFFFFFFh = not counted)
 3Ch DWORD broadcast bytes received (FFFFFFFFh = not counted)
 40h DWORD frames discarded--hardware error (FFFFFFFFh = not counted)
 44h DWORD total frames transmitted (FFFFFFFFh = not counted)
 48h DWORD total bytes transmitted (FFFFFFFFh = not counted)
 4Ch DWORD multicast frames transmitted (FFFFFFFFh = not counted)
 50h DWORD broadcast frames transmitted (FFFFFFFFh = not counted)
 54h DWORD broadcast bytes transmitted (FFFFFFFFh = not counted)
 58h DWORD multicast bytes transmitted (FFFFFFFFh = not counted)
 5Ch DWORD frames not transmitted--timeout (FFFFFFFFh = not counted)
 60h DWORD frames not transmitted--hardware error (FFFFFFFFh = not countd)
 64h N BYTEs vendor-specific info

Bitfields for MAC status bits:
Bit(s) Description (Table 01442)
 0-2 operational status
  000 hardware not installed
  001 hardware failed startup diagnostics
  010 hardware configuration problem
  011 hardware fault
  100 operating marginally due to soft faults

```
  101 reserved
  110 reserved
  111 hardware fully operational
 3  MAC bound
 4  MAC open
 5  diagnostics in progress
 6-31 reserved

Bitfields for packet filter flags:
Bit(s)  Description (Table 01443)
 0  directed/multicast or group/functional
 1  broadcast
 2  promiscuous
 3  all source routing
 4-15 reserved, must be zero
--------D-214402-----------------------------
INT 21 U - MS-DOS 7.0+ - CONFIG$ device - GET CONFIGURATION INFORMATION
  AX = 4402h
  BX = file handle for character device "CONFIG$"
  CX = number of bytes to read (at least 5)
  DS:DX -> buffer for ??? data (see #01444)
Return: CF clear if successful
      AX = number of bytes actually read
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
SeeAlso: AX=4403h"CONFIG$"

Format of MS-DOS 7.0 CONFIG$ ??? data:
Offset  Size  Description (Table 01444)
 00h  WORD  ??? (0000h)
 02h  WORD  ??? (0000h or value read from IO.SYS segment 70h)
 04h  BYTE  ??? (value read from IO.SYS segment 70h)
--------I-214402-----------------------------
INT 21 U - IBM SYSTEM 36/38 WORKSTATION EMULATION - VDI.SYS - GET ???
  AX = 4402h
  BX = handle for character device "GDMS"
  CX = number of bytes to read (>= 4)
  DS:DX -> buffer (see #01445)
Return: CF set on error
      AX = error code (see #01680 at AH=59h/BX=0000h)
  CF clear if successful
```

```
      AX = number of bytes read

Format of System 36/38 emulator returned data:
Offset  Size  Description (Table 01445)
 00h  4 BYTEs ???
 04h  DWORD pointer to ???
 08h  4 BYTEs ???
--------m-214402-----------------------------
INT 21 U - LASTBYTE.SYS v1.19 - IOCTL - GET ??? TABLE
  AX = 4402h
  BX = handle for device "LA$TBYTE"
  CX = 0004h
  DS:DX -> DWORD to hold address of 39-byte table of ???
Return: CF set on error
      AX = error code (see #01680 at AH=59h/BX=0000h)
  CF clear if successful
      AX = number of bytes read
Program: LASTBYTE.SYS is part of the shareware "The Last Byte" memory
    management package by Key Software Products
SeeAlso: AX=4402h"HIGHUMM"
--------m-214402-----------------------------
INT 21 - HIGHUMM.SYS v1.17+ - IOCTL - GET API ADDRESS
  AX = 4402h
  BX = handle for device "KSP$UMM"
  CX = 0004h
  DS:DX -> DWORD to hold entry point (see #01446)
Return: CF set on error
      AX = error code (see #01680 at AH=59h/BX=0000h)
  CF clear if successful
      AX = number of bytes read
Program: HIGHUMM.SYS is part of the shareware "The Last Byte" memory
    management package by Key Software Products
SeeAlso: AX=4402h"LASTBYTE"

(Table 01446)
Call HIGHUMM.SYS entry point with:
  AH = 00h allocate UMB (same as XMS function 10h) (see INT 2F/AX=4310h)
      DX = size in paragraphs
      Return: BX = segment number (if successful)
        DX = size of requested block/size of largest block
  AH = 01h deallocate UMB (same as XMS func 11h) (see INT 2F/AX=4310h)
```

```
      DX = segment number of UMB
  AH = 02h request a bank-switched memory block
      DX = size in paragraphs
      Return: BX = segment number (if successful)
        DX = size of requested block/size of largest block
  AH = 03h release a bank-switched memory block
      DX = segment number
  AH = 04h transfer data to/from high memory
      DS:SI -> source
      ES:DI -> destination
      CX = length in bytes
      Note: enables bank-switched memory, does the copy, then disables
    bank-switched memory
  AH = 05h get a word from bank-switched memory
      ES:DI -> word to read
      Return: DX = word
  AH = 06h put a word to bank-switched memory
      ES:DI -> word to write
      DX = word
  AH = 07h put a byte to bank-switched memory
      ES:DI -> byte to write
      DL = byte
  AH = 08h enable bank-switched memory
      DS:SI -> 6-byte status save area
  AH = 09h disable bank-switched memory
      DS:SI -> 6-byte save area from enable call (AH=08h)
  AH = 0Ah assign name to UMB or high bank-switched block
      DX = segment number
      DS:SI -> 8-byte blank-padded name
  AH = 0Bh locate UMB block by name
      DS:SI -> 8-byte blank-padded name
      Return: BX = segment number (if successful)
        DX = size of block
  AH = 0Ch locate bank-switched block by name
      DS:SI -> 8-byte blank-padded name
      Return: BX = segment number (if successful)
        DX = size of block
Return: AX = status code
      0001h successful
      0000h failed
    BL = error code
```

```
        80h not implemented
        B0h insufficient memory, smaller block available
        B1h insufficient memory, no blocks available
        B2h invalid segment number
Note: only functions 00h and 01h are always available; the remaining
    functions are only enabled if the proper commandline switch is given
--------c-214402-----------------------------
INT 21 - SMARTDRV.SYS v3.x only - IOCTL - GET CACHE STATUS
  AX = 4402h
  BX = file handle for device "SMARTAAR"
  CX = number of bytes to read (min 28h)
  DS:DX -> buffer for status record (see #01447)
Return: CF clear if successful
      AX = number of bytes actually read
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Note: NCache2 (from the Norton Utilities v8.0) attempts to support this
    interface, but does not do so correctly, often hanging the system;
    one should use the SmartDrive v4.x or NCache private interfaces
    (see INT 2F/AX=4A10h/BX=0000h,INT 2F/AX=FE00h/DI=4E55h)
SeeAlso: AX=4403h"SMARTDRV",INT 2F/AX=4A10h/BX=0000h

Format of SMARTDRV status record:
Offset  Size  Description (Table 01447)
 00h  BYTE  write-through flag (always 01h)
 01h  BYTE  writes should be buffered (always 00h)
 02h  BYTE  cache enabled if 01h
 03h  BYTE  driver type (01h extended memory, 02h expanded)
 04h  WORD  clock ticks between cache flushes (currently unused)
 06h  BYTE  cache contains locked tracks if nonzero
 07h  BYTE  flush cache on INT 19 reboot if nonzero
 08h  BYTE  cache full track writes if nonzero
 09h  BYTE  double buffering (for VDS) state (00h off, 01h on, 02h dynamic)
 0Ah  DWORD original INT 13 vector
 0Eh  BYTE  minor version number
 0Fh  BYTE  major version number
 10h  WORD  unused
 12h  WORD  sectors read      \
 14h  WORD  sectors already in cache   > may be scaled rather than
 16h  WORD  sectors already in track buffer /  absolute counts
 18h  BYTE  cache hit rate in percent
```

```
 19h  BYTE  track buffer hit rate in percent
 1Ah  WORD  total tracks in cache
 1Ch  WORD  number of tracks in use
 1Eh  WORD  number of locked tracks
 20h  WORD  number of dirty tracks
 22h  WORD  current cache size in 16K pages
 24h  WORD  original (maximum) cache size in 16K pages
 26h  WORD  minimum cache size in 16K pages
 28h  DWORD pointer to byte flag to increment for locking cache contents
--------d-214402-----------------------------
INT 21 - CD-ROM device driver - IOCTL INPUT
  AX = 4402h
  BX = file handle referencing character device for CD-ROM driver
  CX = number of bytes to read
  DS:DX -> control block (see #01449)
Return: CF clear if successful
      AX = number of bytes actually read
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Notes:  the data returned depends on the first byte of the control block
    (two bytes for functions 01h/07h/0Bh, see #01449); the remainder of
    the control block is filled by the driver
  some device drivers support several subunits (that is more than one
    drive) but it is not possible to distinguish between them with this
    function; use INT 2F/AX=1510h or INT 2F/AX=0802h instead
  under Windows95, the "mscd$$$$" device cannot be opened so you cannot
    obtain the handle required by this function: use INT 2F/AX=1510h
    or INT 2F/AX=0802h instead
SeeAlso: AX=4403h"CD-ROM",INT 2F/AX=0802h,INT 2F/AX=1510h

(Table 01448)
Values for CD-ROM data being requested:
 00h  device driver header address
 01h  drive head location
 02h  reserved
 03h  error statistics
 04h  audio channel info
 05h  raw drive bytes (uninterpreted and device-specific)
 06h  device status
 07h  sector size
 08h  volume size
```

```
 09h  media change status
 0Ah  audio disk info
 0Bh  audio track info
 0Ch  audio Q-Channel info
 0Dh  audio sub-channel info
 0Eh  UPC code
 0Fh  audio status info

Format of CD-ROM control block:
Offset  Size  Description (Table 01449)
 00h  BYTE  data being requested (see #01448)
---function 00h---
 01h  DWORD device driver header address (see also AH=52h,#01646)
---function 01h---
 01h  BYTE  (call) addressing mode
    00h HSG
    01h Red Book
 02h  DWORD current location of drive's head
    logical sector number in HSG mode
    frame/second/minute/unused in Red Book mode
    (HSG sector = minute * 4500 + second * 75 + frame - 150)
---function 03h---
 01h  N BYTEs undefined as of 5 Aug 88 specification
---function 04h---
 01h  BYTE  input channel (0-3) for output channel 0
 02h  BYTE  volume for output channel 0
 03h  BYTE  input channel (0-3) for output channel 1
 04h  BYTE  volume for output channel 1
 05h  BYTE  input channel (0-3) for output channel 2
 06h  BYTE  volume for output channel 2
 07h  BYTE  input channel (0-3) for output channel 3
 08h  BYTE  volume for output channel 3
Notes:  output channels 0 and 1 are left and right, 2 and 3 are left prime and
    right prime; a volume of 00h is off
  the default setting is for each input channel to be assigned to the
    same-numbered output channel at full (FFh) volume
---function 05h---
 01h  BYTE  number of bytes read
 02h 128 BYTEs  buffer for drive bytes
---function 06h---
 01h  DWORD device parameters (see #01450)
```

```
---function 07h---
 01h  BYTE  (call) read mode
    00h cooked
    01h raw
 02h  WORD  (return) sector size in bytes
---function 08h---
 01h  DWORD volume size in sectors
BUGS: Aztech device driver v1.10 and v1.27 (at least) return the last sector
    number, i.e. total number of sectors - 1
  Windows95 returns the total number of sectors + 150 (see the Red Book
    to HSG conversion formula at function 01h to understand why this
    happens)
---function 09h---
 01h  BYTE  media change status
    00h don't know
    01h media unchanged
    FFh media has been changed
---function 0Ah---
 01h  BYTE  lowest audio track number
 02h  BYTE  highest audio track number
 03h  DWORD start address of lead-out track (Red Book format)
--function 0Bh---
 01h  BYTE  (call) track number
 02h  DWORD starting point of track (Red Book format)
 06h  BYTE  track control info
    bits 15,14,12: track type (notice: bits not contiguous!)
      000 two audio channels, no pre-emphasis
      001 two audio channels with pre-emphasis
      010 data track
      100 four audio channels, no pre-emphasis
      101 four audio channels with pre-emphasis
      other reserved
    bit 13: digital copy permitted
---function 0Ch---
 01h  BYTE  CONTROL and ADR byte (as received from drive)
 02h  BYTE  track number
 03h  BYTE  point or index
 04h  BYTE  minute  \
 05h  BYTE  second   > running time within track
 06h  BYTE  frame /
 07h  BYTE  zero
```

```
 08h  BYTE  "AMIN" or "PMIN"     \
 09h  BYTE  "ASEC" or "PSEC"      > running time on disk
 0Ah  BYTE  "AFRAME" or "PFRAME" /
---function 0Dh---
 01h  DWORD starting frame address (Red Book format)
 05h  DWORD transfer address
 09h  DWORD number of sectors to read
Note: copies 96 bytes of sub-channel info per sector into buffer
---function 0Eh---
 01h  BYTE  CONTROL and ADR byte
 02h  7 BYTEs UPC/EAN code (13 BCD digits,low-order nybble of last byte is 0)
 09h  BYTE  zero
 0Ah  BYTE  "AFRAME"
---function 0Fh---
 ??? documentation not yet available
 01h  WORD  pause status (0000h not paused, 0001h paused)
 03h  DWORD audio play start address
 07h  DWORD ??? audio play length or end address

Bitfields for CD-ROM device parameters:
Bit(s)  Description (Table 01450)
 0  door open
 1  door unlocked
 2  supports raw reading in addition to cooked
 3  writable
 4  can play audio/video tracks
 5  supports interleaving
 6  reserved
 7  supports prefetch requests
 8  supports audio channel control
 9  supports Red Book addressing in addition to HSG
 10 audio is playing
 11 no disk in drive
 12 supports R-W subchannels
--------m-214402-----------------------------
INT 21 - Quarterdeck - QEMM-386 v5+ - GET API ENTRY POINT
  AX = 4402h
  BX = file handle for device "QEMM386$"
  CX = 0004h
  DS:DX -> DWORD buffer for API entry point
Return: CF clear if successful
```

```
      buffer filled (refer to INT 67/AH=3Fh for entry point parameters)
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Note: Quarterdeck recently (June 1993) documented this function, but the
    documentation incorrectly states that it is only available for
    QEMM 6+
SeeAlso: AX=4402h"HOOKROM",INT 2F/AX=D201h/BX=5145h,INT 67/AH=3Fh
--------Q-214402-----------------------------
INT 21 U - Quarterdeck - HOOKROM.SYS - GET HOOKED VECTOR TABLE
  AX = 4402h
  BX = file handle for device "HOOKROM$"
  CX = 0004h
  DS:DX -> DWORD buffer for address of hooked vector table (see #01451)
Return: CF clear if successful
      DS:DX buffer filled
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
SeeAlso: AX=4402h/SF=01h

Format of HOOKROM.SYS hooked vector table entry:
Offset  Size  Description (Table 01451)
 00h  5 BYTEs FAR jump to actual interrupt handler
    (end of table if first byte is not EAh)
 05h  BYTE  interrupt vector number
--------d-214402-----------------------------
INT 21 - Advanced SCSI Programming Interface (ASPI) - INTERFACE
  AX = 4402h
  BX = file handle for device "SCSIMGR$"
  CX = 0004h or 0005h (refer to notes below)
  DS:DX -> buffer for result (see #01452), set to zeros before call
Return: CF clear if successful
      AX = 0004h or 0005h (refer to notes below)
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Notes:  the variant of the call requesting five bytes is an UNDOCUMENTED
    extension supported by Adaptec's ASPI2DOS.SYS, ASPI4DOS.SYS, and
    ASPI7DOS.SYS; if made of a host manager which does not support the
    variant, only four bytes will be returned.  If the variant is
    supported, Adaptec's WINASPI.DLL assumes that the host manager is
    an "advanced" one which operates in either real or protected mode
    (and thus does not require a DPMI INT 31/AX=0301h call to be invoked
```

```
    from protected mode).  Support of the five-byte variant also appears
    to imply that an advanced ASPI host manager uses no temporary
    storage space except the SRB (see #01454) and the stack, and that it
    is fully reentrant.
  if called with a standard request for four bytes, even Adaptec's
    advanced drivers return only the requested four bytes containing the
    ASPI entry point address
  the function address is called with the address of a SCSI Request
    Block (see #01454) on the stack and the caller must clean up the
    stack
SeeAlso: AX=440Ch"ASPITAPE",INT 11/AH=FFh"WD7000"

Format of ASPI IOCTL result:
Offset  Size  Description (Table 01452)
 00h  DWORD function address
 04h  BYTE  number of SCSI host adapters supported by host manager

(Table 01453)
Values for ASPI request number:
 00h  "HA_INQ"  host adapter inquiry / extended host adapter inquiry
 01h  "GET_TYPE"  get device type
 02h  "EXEC_SIO"  execute SCSI I/O
 03h  "ABORT_SRB" abort SRB
 04h  "RESET_DEV" reset SCSI device
 05h  "SET_HAPRMS"  set host adapter parameters
 06h  get disk drive information
 7Fh  (Adaptec) get ASPI manager info?
Note: request 7Fh is supported by all the Adaptec's DOS ASPI managers:
    ASPI2DOS.SYS (for AHA-152x), ASPI4DOS.SYS (for AHA-154x/164x),
    ASPI7DOS.SYS (for AIC-77xx), and ASPI8DOS.SYS (for AIC-78xx)

Format of SCSI Request Block (64 bytes):
Offset  Size  Description (Table 01454)
 00h  BYTE  request number (see #01453)
 01h  BYTE  request status (see #01455)
 02h  BYTE  host adapter ID
 03h  BYTE  request flags (see #01456)
 04h  DWORD reserved (0)
---request 00h---
 08h  BYTE  (ret) number of host adapters
 09h  BYTE  (ret) target adapter ID
```

```
 0Ah 16 BYTEs (ret) SCSI manager ID
 1Ah 16 BYTEs (ret) host adapter ID
 2Ah 16 BYTEs (ret) host adapter-unique parameters (see #90010,#90011)
---extended request 00h---
 04h  2 BYTEs (call) extended request signature 55h AAh
    (ret) if extended request supported, reply AAh 55h
 06h  WORD  (call) length of extended buffer at offset 3Ah
    (ret) number of bytes returned in extended buffer
 08h  BYTE  (ret) number of host adapters
 09h  BYTE  (ret) target adapter ID
 0Ah 16 BYTEs (ret) SCSI manager ID
 1Ah 16 BYTEs (ret) host adapter ID
 2Ah 16 BYTEs (ret) host adapter-unique parameters
 3Ah  N BYTEs extended buffer
    3Ah WORD  features (see #01457)
    3Ch WORD  maximum scatter/gather list length
    3Eh DWORD maximum SCIS data transfer size (0 = unlimited)
---request 01h---
 08h  BYTE  target ID
 09h  BYTE  logical unit number
 0Ah  BYTE  (ret) device type (see #01460)
---request 02h---
 08h  BYTE  target ID
 09h  BYTE  logical unit number
 0Ah  DWORD data allocation length
    (ret) residual byte length (if supported and requested)
 0Eh  BYTE  sense allocation length
 0Fh  DWORD data buffer pointer
 13h  DWORD next request pointer (for linking)
 17h  BYTE  CDB length
 18h  BYTE  (ret) host adapter status (see #01454)
 19h  BYTE  (ret) target status (see #01459)
 1Ah  DWORD post routine address (see #01466)
 1Eh  WORD  real mode Post DS
 20h  DWORD SRB pointer
 24h  WORD  reserved
 26h  DWORD SRB physical address
 2Ah 22 BYTEs SCSIMGR$ workspace
 40h  N BYTEs CCB, including sense data (20-24 bytes)
---request 03h---
 08h  DWORD address of SRB to abort
```

```
---request 04h---
 08h  BYTE  target ID
 09h  BYTE  logical unit number
 0Ah 14 BYTEs reserved
 18h  BYTE  (ret) host adapter status (see #01458)
 19h  BYTE  (ret) target status (see #01459)
 1Ah  DWORD post routine address
 1Eh 34 BYTEs workspace
---request 05h---
 08h 16 BYTEs host adapter-unique parameters
---request 06h---
 08h  BYTE  target ID
 09h  BYTE  logical unit number
 0Ah  BYTE  disk drive flags (see #01461)
 0Bh  BYTE  INT 13h drive number
 0Ch  BYTE  preferred head number translation
 0Dh  BYTE  preferred sector size translation
 0Eh 10 BYTEs reserved
---request 7Fh---
 08h  WORD  base I/O port address
 0Ah  BYTE  number of I/O ports used
 0Bh  BYTE  ??? (01h returned for AHA-152x)
 0Ch  BYTE  interrupt level
 0Dh  BYTE  DMA channel
 0Eh  BYTE  (ASPI7DOS.SYS) EISA slot  number
    (ASPI8DOS.SYS) PCI device number
 0Fh  BYTE  bits 7-1: reserved (0)
    bit 0: (AHA-152x) ???
 10h  DWORD ASPI entry point address
 14h  DWORD previous ASPI entry point address
 18h  WORD  offset to "ASPI request dispatcher" procedure
 1Ah  WORD  offset to "interrupt handler" procedure
 1Ch  WORD  offset to some procedure
 1Eh  WORD  offset to some procedure
 20h  WORD  offset to host adapter data
 22h  BYTE  ??? (apparently always 02h)
 23h  BYTE  reserved??? (0)
 24h  BYTE  (ASPI8DOS.SYS) PCI bus number?
 25h  11 BYTEs  reserved??? (0)
SeeAlso: #01462
```

```
(Table 01455)
Values for ASPI request status:
 00h  not done yet
 01h  completed successfully
 02h  aborted by host
 04h  SCSI I/O error
 80h  invalid
 81h  no adapter
 82h  no device attached
 else status
SeeAlso: #01454

Bitfields for ASPI request flags:
Bit(s)  Description (Table 01456)
 0  posting enabled
 1  linking enabled
 2  residual byte length reported in Data Allocation Length field
 3  transfer from SCSI target to host
 4  transfer from host to SCSI target
 5  scatter/gather
 7-6  reserved
Note: no data is transferred if both bits 3 and 4 are set; if neither is
    set, the direction is determined by the SCSI command
SeeAlso: #01454

Bitfields for ASPI extended features:
Bit(s)  Description (Table 01457)
 0  scatter/gather supported
 1  residual byte length reported
 2  Wide SCSI 16 host adapter
 3  Wide SCSI 32 host adapter
 15-4 reserved
SeeAlso: #01454

(Table 01458)
Values for host adapter status:
 00h  no error detected
 11h  select timeout
 12h  data overrun
 13h  bus error
 14h  bus failure
```

```
(Table 01459)
Values for target status:
 00h  no status
 02h  sense data stored in SRB
 08h  target busy
 18h  reservation error

(Table 01460)
Values for device type:
 00h  disk drive
 01h  tape drive (streamer)
 02h  printer
 03h  processor
 04h  WORM drive
 05h  CD-ROM drive
 06h  scanner
 07h  optical drive
 08h  autochanger
 09h  communications device

(Table 01461)
Values for disk drive flags:
 00h  no INT 13 access
 01h  INT 13 with DOS access
 02h  INT 13 without DOS access
 03h  invalid flags

Format of CCB:
Offset  Size  Description (Table 01462)
 00h  BYTE  command code (see #01463)
 01h  BYTE  flags
    bits 4-0: vary by function
    bits 7-5: logical unit number
 02h  BYTE  "adr_1"
 03h  BYTE  "adr_0"
 04h  BYTE  length
 05h  BYTE  control
  ...
 06h/0Ah 14 BYTEs buffer for sense data (see #01464)
SeeAlso: #01454
```

```
(Table 01463)
Values for CCB command code:
 00h  test unit ready
 01h  rewind
 03h  request sense data
 05h  get block size limits
 08h  Group 0 read
 0Ah  Group 0 write
 10h  write file marks
 11h  SCSI Space (set position?)
 12h  SCSI Inquire
 15h  set mode information
 16h  reserve SCSI device
 17h  release SCSI device
 19h  erase
 1Ah  request mode information
 1Bh  load/unload media
 1Dh  request target self-check
 24h  set window parameters
 25h  get window parameters
 28h  Group 1 read
 2Ah  Group 1 write
 31h  document feeder control
 34h  get scan data status
---vendor-specific commands---
 D3h  get document feeder status
 D4h  set document feeder mode

Format of sense data:
Offset  Size  Description (Table 01464)
 00h  BYTE  error code (bit 7 set if valid)
 01h  BYTE  segment number
 02h  BYTE  sense key
    bit 6: EOM
    bit 5: ILI
    bits 0-3: sense key (see #01465)
 03h  4 BYTEs information bytes
 07h  BYTE  additional sense length (0Ah)
 08h  4 BYTEs command-specific information
 0Ch  BYTE  additional sense code
```

```
 0Dh  BYTE  additional sense code qualifier
 0Eh  BYTE  field replaceable unit code
 0Fh  3 BYTEs sense key specific bytes

(Table 01465)
Values for sense key:
 00h  no sense data
 02h  SCSI unit not ready
 03h  media error
 04h  unrecoverable hardware error
 05h  illegal parameter in CDB
 06h  target has been reset
 0Bh  target aborted command

(Table 01466)
Values ASPI post function is called with:
  STACK:  DWORD -> SRB (see #01454) which completed
  interrupts disabled
Return: EBP, EBX, ESI, EDI must be preserved
  interrupts disabled
Note: the post function may issue any ASPI function except an abort; it
    should complete as quickly as possible

Format of ASPI2DOS.SYS v3.65 host adapter unique parameters:
Offset  Size  Description (Table 90010)
 00h  WORD  reserved? (0)
 02h  WORD  reserved? (0)
 04h  WORD  base I/O port address
 06h  BYTE  interrupt level
 07h  9 BYTEs reserved  (0)
SeeAlso: #90011,#01454

Format of ASPI4DOS.SYS v3.35 host adapter unique parameters:
Offset  Size  Description (Table 90011)
 00h  WORD  offset to "ASPI request dispatcher" procedure
 02h  WORD  offset to "interrupt handler" procedure
 04h 12 BYTEs reserved  (0)
SeeAlso: #90010,#01454
--------m-214402-----------------------------
INT 21 U - Qualitas 386MAX v6.00+ - IOCTL INPUT - GET STATE
  AX = 4402h
```

```
  BX = file handle for device "386MAX$$"
  CX = number of bytes to read
  DS:DX -> BYTE 03h followed by 386MAX state buffer (see #01467)
Return: CF clear if successful
      buffer at DS:DX+1 filled
      AX = number of bytes actually copied
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Notes:  if the value given in CX is less than the size of the state record
    (5Ah for v6.01, 66h for v7.00), only a partial state record will be
    returned
  the state is 40h bytes for 386MAX (actually ASTEMM) v2.20 ("386MAX$$"
    did not exist yet, use "QMMXXXX0" and then "EMMXXXX0" instead) and
    56h bytes for v5.11.
  to invoke 386MAX API functions, push DX onto the stack, load DX with
    the word at offset 25h in the returned state, load all other
    registers as needed for the desired function, and execute an
    OUT DX,AL or OUT DX,AX; DX will be set to the pushed value on return
    if it is not otherwise modified by the API function.  For safety,
    in case a function is not supported or 386MAX is not present, SP
    should be saved and restored around the API call.
  Windows 3.1 Standard mode, LAN Manager, and Windows for Workgroups all
    use the 386MAX API; LAN Manager and Windows for Workgroups reportedly
    make some calls incorrectly
SeeAlso: AX=4403h/SF=03h"386MAX",INT 67/AH=3Fh

Format of 386MAX v6.01+ state:
Offset  Size  Description (Table 01467)
 -1 BYTE  (call) 03h
 00h  6 BYTEs signature "386MAX"
 06h  4 BYTEs version string "N;NN" or "N.NN" (i.e. "6;01" for v6.01)
    (';' by default; apparently changed to a period when 386MAX
    has linked high RAM into DOS's memory chain)
 0Ah  WORD  segment of low-memory portion of 386MAX.SYS
 0Ch  2 BYTEs ???
 0Eh  WORD  segment of ??? memory block or 0000h
 10h  WORD  bit flags 1 (see #01468)
 12h  WORD  starting address of video memory in KB
 14h  2 BYTEs ???
 16h  WORD  total high DOS memory in KB
 18h  2 BYTEs ???
```

```
 1Ah  WORD  available shared memory in KB
 1Ch  WORD  KBytes extended memory used by 386MAX
 1Eh  2 BYTEs ???
 20h  WORD  total extended memory in KB
 22h  WORD  IO port to write (OUT DX,AL) to invoke 386MAX INT 15 functions
 24h  WORD  IO port to write (OUT DX,AL) to invoke 386MAX API functions
      (see #01481)
 26h  WORD  ??? (depends on DOS version)
 28h  WORD  size of ??? in paragraphs
 2Ah  DWORD machine type (see #01476)
 2Eh  DWORD -> first DOS memory control block
 32h  WORD  system configuration flags (see #01469)
 34h  WORD  debugging flags 1 (see #01470)
 36h  WORD  debugging flags 2 (see #01471)
 38h  2 BYTEs ???
 3Ah  WORD  segment of first MCB in high memory chain
 3Ch  WORD  feature flags 1 (see #01473)
 3Eh  WORD  feature flags 2 (see #01474)
 40h  WORD  feature flags 3 (see #01475)
 42h  WORD  segment of first 386MAX control block??? (see #01477)
 44h  WORD  amount of memory to report available on INT 15/AH=88h
 46h  4 BYTEs ???
 4Ah  WORD  number of K at start of address space swapped with fast
      extended memory (SWAP= parameter)
 4Ch  2 BYTEs ???
 4Eh  WORD  segment address of ???
 50h  WORD  debugging flags 3 (see #01472)
 52h  DWORD old INT 21h
 56h  DWORD pointer to 386MAX's EMS (INT 67h) handler
---386MAX v7.00---
 5Ah  DWORD KB of extended memory managed by 386MAX
 5Eh  DWORD bytes of extended memory (EXT= parameter)
 62h  4 BYTEs ???

Bitfields for 386MAX bit flags 1:
Bit(s)  Description (Table 01468)
 1  ???
 2  allow A20 to be enabled/disabled???
 3  ??? (cleared by calling INT 67 functions or starting MSWindows)
 4  high RAM present???
 5  386MAX in AUTO mode
```

```
 6  386MAX enabled
 7  386MAX is providing EMS services
 8  ??? (affects API function 08h)
 9  A20 gate closed (A20 disabled) (see INT 15/AX=2402h)
 10 Weitek support enabled
 11 ???
 12 ROMs not shadowed???
 13 QPMS has been used
 14 ???
 15 ???

Bitfields for 386MAX system configuration flags:
Bit(s)  Description (Table 01469)
 1  ROM compressed???
 3  ???
 5  386MAX loaded into high memory
 6  Microchannel bus
 7  Weitek math coprocessor detected
 9  ??? (also generates INT 01 on ??? and INT 03 on ???)
 11 PC/XT (thus only single 8259 interrupt controller present, DMA only
    in 1st megabyte, etc)
 13 LMLTOP= specified
 14 enable A20 control???
 15 ???

Bitfields for 386MAX debugging flags 1:
Bit(s)  Description (Table 01470)
 0  DEBUG=LED
 1  DEBUG=X67
 2  DEBUG=INV
 3  DEBUG=EMSPTED
 4  DEBUG=JMP
 5  DEBUG=CALL
 6  DEBUG=HLT
 7  DEBUG=PMR
 8  DEBUG=CR3
 9  DEBUG=CAPS or DEBUG=INT
 10 DEBUG=RC
 11 DEBUG=ROM
 12 DEBUG=XM
 13 DEBUG=SOR
```

```
 14 DEBUG=XR
 15 DEBUG=EMSERR (generate INT 01 on returning error from EMS call)

Bitfields for 386MAX debugging flags 2:
Bit(s)  Description (Table 01471)
 0  DEBUG=ROMSWAP
 1  DEBUG=UNSHADOWROM
 2  DEBUG=COMPROM
 3  DEBUG=DPMIPHYS
 4  DEBUG=ALLROM
 5  DEBUG=VMS
 6  DEBUG=XMS (generate INT 01 on XMS calls)
 7  DEBUG=I06
 8  DEBUG=VCPI
 9  DEBUG=XDMA
 10 DEBUG=X09
 13 DEBUG=I67 (generate INT 01 on every INT 67 call)
 14 DEBUG=EVM (generate INT 01 on entering V86 mode)
 15 DEBUG=EMSSAVE or DEBUG=VDS

Bitfields for 386MAX debugging flags 3:
Bit(s)  Description (Table 01472)
 10 DEBUG=EPM
 12 DEBUG=ABIOS
 13 DEBUG=XMSPTED
 14 DEBUG=TIME
 15 DEBUG=SCRUB

Bitfields for 386MAX feature flags 1:
Bit(s)  Description (Table 01473)
 1  Weitek present
 2  no DPMI services
 3  NODMA
 4  TERSE
 5  NOROM
 6  NOPARITY
 8  NOFLEX (IGNOREFLEXFRAME)
 11 don't create UMBs
 12 don't backfill below video memory (NOLOW)
 13 FRAME= specified
 14 EXT= specified
```

```
 15 NOEMS, allow prior expanded memory manager to provide EMS

Bitfields for 386MAX feature flags 2:
Bit(s)  Description (Table 01474)
 0  UNSHIFT specified (FORCEA20 disabled)
 1  NOXRAM
 2  NOSCSI specified
 3  SCREEN specified
 4  enabled EISADMA
 5  slow DMA
 6  RESETKEYB specified
 7  ???
 9  TOP384
 10 ???
 11 NOWARMBOOT
 12 USE= specified
 13 ROM= specified

Bitfields for 386MAX feature flags 3:
Bit(s)  Description (Table 01475)
 0  Windows3 support enabled
 1  SHADOWROM
 2  don't compress ROM (NOCOMPROM)
 3  ??? (related to PRGREG=)
 4  ??? (related to PRGREG=)
 5  SHADOWRAM
 6  DOS4 specified
 7  NOLOADHIGH
 8  NOPULSE
 11 FORCEA20
 12 DMA buffer enabled
 13 NOSCRUB
 15 NOFRAME

Bitfields for 386MAX machine type:
Bit(s)  Description (Table 01476)
 12 Amstrad
 13 Epson
 14 Zenith Data Systems
 15 "ASEM"
 16 NEC
```

```
 17 "HPRS" model codes 69h and 6Ah
 18 Dell
 19 "CA"
 20 ITT (Xtra Business Systems/Alcatel)
 21 Toshiba 5100
 22 Olivetti
 23 Quadram Quad386 (BIOS model FEh, submodel A6h)
 24 Tandy???
 25 AST 386
 26 INBOARD, ??? version
 27 INBOARD, ??? version
 28 INBOARD, ??? version
 29 "HPRS"
 30 Compaq 386
 31 JET386

Format of 386MAX control block:
Offset  Size  Description (Table 01477)
 00h  WORD  segment of next block (FFFFh if last)
 02h  WORD  segment of previous block (FFFFh if first)
 04h 12 BYTEs filename
 10h  WORD  resident size in paragraphs
 12h  WORD  environment size???
 14h  WORD  real prsent environment size + 1 (0000h if ENVSAVE used)
 16h  2 BYTEs ???
 18h  DWORD initial size or SIZE=n in 386LOAD commandline
 1Ch  DWORD SIZE=-1 ???
 20h  DWORD SIZE= ???
 24h  BYTE  PRGREG= if specified, else FFh
 25h  BYTE  ENVREG= if specified, else FFh
 26h  BYTE  FlexFrame (00h not present, 01h present)
 27h  3 BYTEs ???
 2Ah  BYTE  GROUP= or 00h if not present
 2Bh  BYTE  ???
 2Ch  WORD  PSP

Format of 386MAX high memory info record:
Offset  Size  Description (Table 01478)
 00h  WORD  segment address of memory region
 02h  WORD  size of memory region in paragraphs
 04h  BYTE  type or flags???
```

```
    00h if locked out
    02h if EMS page frame
    04h if high RAM
    42h if ROM
 05h  BYTE  ???

Format of 386MAX ROM shadowing record:
Offset  Size  Description (Table 01479)
 00h  WORD  logical start segment of ROM??? (may be used by BlueMAX when it
      squeezes together the ROMs to make room)
 02h  WORD  physical start segment of ROM
 04h  2 BYTEs ???
 06h  WORD  size of shadowed ROM in paragraphs
 08h  2 BYTEs ???
 0Ah  WORD  flags
    bit 15: shadowing enabled for this ROM???
    bit 14: ???
    bit 13: ???
    bit 12: ???
    bit 10: ???

(Table 01480)
Values for 386MAX memory type:
 00h  unused by EMS
 01h  DOS
 04h  page frame overlapping ROM???
 80h  high memory
 84h  page frame???
 87h  video ROM???
Note: the type may be 00h (unused) if the 16K page is split among different
    uses (such as ROM and high RAM)

(Table 01481)
Call 386MAX API (via OUT DX,AL) with:
  STACK: WORD value for DX
  AH = 00h unused
    Return: AH = 84h (unsupported function)
  AH = 01h get high memory information
    ES:DI -> buffer for array of high memory info records
      (see #01478)
    Return: CX = number of records placed in buffer
```

```
AH = 02h get shadowed ROM info
  ES:DI -> buffer for array of ROM shadowing records (see #01479)
  Return: CX = number of records placed in buffer
AH = 03h get 386MAX state
  ES:DI -> 90-byte buffer for state (see #01467)
  Return: AH = 00h (successful)
    buffer filled
AH = 04h get memory types???
  ES:DI -> buffer for memory type info (array of bytes, one per
    16K page) (see #01480)
  Return: CX = number of bytes placed in buffer
AH = 05h get page table entries
  AL = A20 control (00h enable A20 first, 01h leave unchanged)
  CX = buffer size in bytes (0000h = enough for all memory from
    given start to end of memory managed by 386MAX)
  SI = first K to report (rounded down to 4K page)
  ES:DI -> buffer for returned page table entries
  Return: CX = number of bytes returned (four per 4K page)
    ES:DI buffer filled
AH = 06h get memory speed info
  ES:DI -> buffer for memory speed records (see #01482)
  Return: AH = 00h (successful)
    CX = number of bytes placed in buffer
  Note: this function can take over a second to execute
AH = 07h map/unmap multiple handle pages
  DX = EMS handle (on stack)
  STACK: DWORD -> EMS mapping record
  Return: AH = status (00h,80h,83h,8Ah,8Bh)
  Format of EMS mapping record:
  Offset  Size  Description
   00h  WORD  function
      0000h use physical page numbers
      0001h use segment addresses
   02h  WORD  EMS handle
   04h  WORD  number of mapping entries following
   06h 2N WORDs logical page number and physical page/segment
      logical page FFFFh means unmap physical page
  SeeAlso: INT 67/AH=50h
AH = 08h "EMM2_GOREAL" check whether possible to disable 386MAX
  AL = ??? (00h or nonzero)
  Return: AH = status (00h OK, A4h not possible at this time)
```

```
    Note: if AL=00h, this function always returns success
  AH = 09h toggle Bit Flags 1 flags
    BX = bitmask of bit flags 1's flags to toggle (see #01468)
    Return: AH = 00h (successful)
    Note: enables A20 first
  AH = 0Ah toggle Debugging Flags 1 flags
    BX = bitmask of Debugging Flags 1's bits to toggle (see #01470)
    Return: AH = 00h (successful)
    Notes:  enables A20 first
      does ??? if bit 3 on after specified bits are toggled
  AH = 0Bh toggle Debugging Flags 2 flags
    BX = bitmask of Debugging Flags 2's bits to toggle (see #01471)
    Return: AH = 00h (successful)
    Note: enables A20 first
  AH = 0Ch toggle feature flags 3
    BX = bitmask of feature flags 3's bits to toggle (see #01475)
    Return: AH = 00h (successful)
    Note: enables A20 first
  AH = 0Dh specify 386MAX high-memory location
    BX = segment address of high-memory real-mode portion of 386MAX
    CX = current segment of real-mode stub???
    Return: AH = status (00h successful)
      ???
  AH = 0Eh CRT controller register virtualization
    AL = subfunction
         00h allow access to CRTC I/O ports 03B4h/03B5h, 03D4h/03D5h
         01h trap accesses to CRTC I/O ports
  AH = 0Fh reboot system
    Return: never
  AH = 10h unused
    Return: AH = 84h (unsupported function)
  AH = 11h get high memory information
    ES:DI -> 96-byte buffer for high memory info
    Return: AH = 00h (successful)
      ES:DI buffer filled
    Notes:  each byte in buffer contains bit flags for a 4K page in
        the A000h-FFFFh region
        bit 0: page is writeable
        bit 1: physical address same as linear address
        bit 2: EMS page frame
        bit 6: page is part of the QPMS window
```

```
      this function can take over a second to execute,
        because it does a 128K read for each page in an
        attempt to flush any RAM cache the system may have
AH = 12h shadow RAM mapping
  AL = subfunction
       00h unshadow ROMs (except page FFh if NOWARMBOOT set)
       01h map shadow RAM into ROM regions???
  Return: AH = 00h (successful) if AL=00h or 01h
    AH = 8Fh otherwise
AH = 13h shadow RAM page protection
  AL = subfunction
       00h set all shadowed ROM 4K pages to read-only
       01h set all shadowed ROM 4K pages to read-write
  Return: AH = 00h (successful) if AL=00h or 01h
    AH = 8Fh otherwise
AH = 14h get Programmable Option Select info???
  ES:DI -> 54-byte buffer for POS data???
  Return: AH = 00h if successful
    AH = A4h on error
  Note: the buffer consists of nine 6-byte fields; the first
       eight for slots 1-8, the last for the system board
AH = 15h ???
  ???
  Return: ???
AH = 16h get 386MAX memory usage screen
  ES:DI -> buffer for memory info display
  CX = size of buffer in bytes
  Return: ES:DI buffer filled with '$'-terminated string (if
       large enough to hold entire usage screen)
  Note: the screen is 0303h bytes in v7.00
AH = 17h Windows 3 startup/termination
  AL = subfunction
       00h Windows3 initializing
    DX (on stack) = Windows startup flags
    DI = Windows version number (major in upper byte)
    ES:BX = 0000h:0000h
    DS:SI = 0000h:0000h
    Return: CX = 0000h if OK for Windows to load
         <> 0 if Windows should not load
      ES:BX -> startup info structure
      DS:SI -> Virtual86 mode enable/disable callback
```

```
      01h Windows3 terminating
    ES:BX -> ???
    DX (on stack) = Windows exit flags
    Return: ???
AH = 18h QPMS (Qualitas Protected Memory Services)
  AL = subfunction
      00h get QPMS configuration
    Return: BX = starting segment of QPMS memory window
      CX = number of 4K pages reserved for QPMS???
      DX = number of 4K pages in QPMS window???
      01h map QPMS memory page???
    BX = 4K page number within memory reserved for QPMS???
    CL = 4K page number within QPMS memory window???
      02h mark all QPMS memory read-only
      03h mark all QPMS memory read-write
  Return: AH = status (00h,8Ah,8Bh,8Fh)
AH = 19h get linear address for physical address
  EDX = physical address (low word on stack)
  Return: AH = status
        00h successful
      EDX = linear address at which physical address
        may be accessed
        8Bh physical address currently not addressable
  Note: enables A20 first
AH = 1Ah set page table entry
  EDX = new page table entry (low word on stack)
  ESI = linear address of page to map (bits 0-11 clear)
  Return: AH = status (00h,8Bh)
  Note: enables A20 first
AH = 1Bh get ???
  Return: AH = status
    BX = ???
    CX = number of ???
    EDX = physical address of ???
AH = 1Ch get original interrupt vector
  AL = interrupt vector (00h-7Fh)
  Return: AH = 00h (successful)
    EDX = original vector before 386MAX loaded (segment in
      high word, offset in low word)
  Note: no range checking is performed; requests for INTs 80h-
      FFh will return random values
```

```
AH = 1Dh display string???
  SI = ???
  Return: AH = 00h (successful)
    ???
  Note: this function appears to be broken in v7.00
AH = 1Eh get memory info
  ES:DI -> memory info (see #01483)
  Return: ???
AH = 1Fh get DPMI host information
  Return: AX = 0000h if successful
    BX = DPMI flags (see #03152 at INT 31/AX=0400h)
    CL = CPU type (02h = 80286, 03h = 80386, etc.)
    DX = DPMI ver supported (DH=major, DL=2-digit minor)
    SI = ???
    ES???:DI -> ???
  Note: NOP if NODPMI switch specified
AH = 20h (v7.00) get ???
    AL = index of ???
    Return: EDX = ??? for specified ???
AH = 21h (v7.00) STACKS support
    AL = 00h get STACKS parameters
  Return: BX = ??? (0060h for v7.00)
    CX = number of stacks for hardware interrupts
    DX = size of each stack in bytes
    SI = ??? (low and high bytes are separate values)
    DI = ??? (low and high bytes are separate values)
        low byte = logical page number set by subfn 02h
    ES = ???
    AL = 01h set ??? "EMM2_DSTKS"
  EBX = ???
  ECX = ???
    AL = 02h set ???
  BL = logical page number for ??? (00h-03h)
  Return: AH = status (00h,8Ah)
AH = 22h (v7.00) call ??? for every load module
    AL = which function to call
  00h call ???
  else call ????
    Return: AH = 00h
    Note: if AL=00h, calls the protected-mode function pointed at by
  the DWORD at offset 22h from the start of each module installed
```

```
    by a LOAD= directive; if AL<>00h, it calls the function
    pointed at by the DWORD at offset 28h of the load module
AH = 23h (v7.00) ???
      AL = 00h set ???
    BL = ???
    Return: AH = 00h or unchanged (depending on ???)
      AL = 01h set ???
    BL = ???
    BH = ???
    CX = ???
    DX = ??? (on top of stack)
    Return: AH = status (00h if successful, 8Fh once table full)
    Note: this call adds one entry to an internal table on each
        call, until the table is full
      AL = 02h get ???
    CX = size of buffer
    ES:DI -> buffer for ??? (60 bytes total data)
    Return: CX = number of bytes actually returned
    Note: returns the array storing the values set with AX=2301h
    Format of one entry in array:
    Offset  Size  Description
     00h  BYTE  ??? (BL from subfn 01h)
     01h  WORD  ??? (CX from subfn 01h)
     03h  BYTE  ??? (BH from subfn 01h)
     04h  WORD  ??? (DX from subfn 01h)
      AL = 03h set ??? name/path
    ES:DI -> buffer containing ASCIZ ???
      AL = 04h get ???
    ES:DI -> buffer for ASCIZ ???
      Note: the ASCIZ string for subfunctions 03h and 04h does not appear
    to be used by 386MAX, and may serve merely for communication
    between two other Qualitas programs
AH = 24h (v7.00) high memory control
      AL = 00h get high memory state
    Return: BX = current state
          00h high memory removed from DOS memory chain
          01h high memory included in DOS memory chain
      AL = 01h set high memory state
    BX = new state
        00h high memory removed from DOS memory chain
        01h high memory included in DOS memory chain
```

```
        else
      Return: ??? (error, but return varies according to ???)
  AH = 25h (v7.00) remove high RAM from DOS memory chain
  AH = 26h (v7.00) ???
      BX = ???
      CX = ???
      SI = ???
      DI = ???
      Return: AH = status
        BX = ???
        CX = ???
  AH = 27h (v7.00) ???
      AL = 00h get ???
    Return: BX = number of paragraphs for ???
      AL = 01h ???
    BX = ???
    ES??? = ???
      AL = 02h ???
    ???
      AL = 03h ???
    CX = ???
    DX = ???
    ES??? = ???
    Return: ???
  AH = 28h (v7.00) get ???
    Return: AH = status (00h,8Fh) (see #03648 at INT 67/AH=40h)
      if AH=00h,
          CX = ???
          DX = ???
  AH = 29h (v7.00) get ???
    Return: AX = ???
  AH = 40h-5Dh EMS services (see INT 67/AH=40h, etc.)
  AH = DEh VCPI services (see INT 67/AX=DE00h, etc.)
Return: AH = status (as for EMS INT 67 calls)
      00h successful
      80h internal error
      81h hardware malfunction
      83h invalid handle
      84h undefined function
      8Ah invalid logical page nuber
      8Bh illegal physical page number
```

```
      8Fh undefined subfunction
      A4h access denied
      etc.
  STACK popped (value placed in DX if no specific return value for DX)

Format of 386MAX memory speed record:
Offset  Size  Description (Table 01482)
 00h  DWORD page table entry for 4K page
 04h  WORD  number of microticks (840ns units) required for REP LODSD of
      entire 4K page

Format of 386MAX memory info [array]:
Offset  Size  Description (Table 01483)
 00h  DWORD linear start address
 04h  DWORD size in bytes
 08h  WORD  XMS handle (if next byte = 04h)
    ??? (if next byte = 05h)
    ??? (if next byte = 06h)
    ??? (if next byte = 13h)
    ??? (if next byte = 14h)
    ??? (if next byte = 15h)
    ??? (if next byte = 23h)
    ??? (if next byte = 24h)
    ??? (if next byte = 26h)
    else unused
 0Ah  BYTE  type
    00h = ???, 01h = VDISK,
    02h = INT 15h extended memory, 03h = ??? extended,
    04h = XMS handle's memory, 05h = ???, 06h = ???, 07h = ???,
    08h = ???, 09h = ???, 0Ah = ???, 0Bh = ???,
    11h = ???, 12h = ???, 14h = ???, 15h = ???,
    19h = ???, 1Ah = ???, 1Bh = ???,
    1Ch = ???, 1Dh = ???, 1Eh = ???, 1Fh = ???,
    20h = ???, 21h = ???, 23h = ???, 24h = ???,
    26h = ???
 0Bh  BYTE  ??? (00h for types 00h-03h, 07h-0Bh, 19h-21h;
         80h for types 04h/13h-15h/23h-26h;
      ??? for type 05h)
--------V-214402-----------------------------
INT 21 - PGS1600.DEV - IOCTL - GET CONFIGURATION INFO
  AX = 4402h
```

```
  BX = file handle for device "PGS1600$"
  CX = 0018h (size of buffer)
  DS:DX -> configuration buffer (see #01484)
Return: CF clear if successful
      buffer filled
      AX = number of bytes actually copied
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Program: PGS1600.DEV is a device driver for the Cornerstone Technology PG1600
    display adapter, which provides a 1600x1200 monochrome display as
    well as one of two emulations, MDA or CGA.
SeeAlso: AX=4403h"PGS1600"

Format of PGS1600.DEV configuration information:
Offset  Size  Description (Table 01484)
 00h  WORD  version (high byte = major, low byte = minor)
 02h  WORD  board initialisation mode
 04h  WORD  board I/O address
    03D0h CGA emulation
    03B0h MDA emulation
    0390h no emulation
    0350h no emulation, alternate
 06h  WORD  emulation buffer segment
    B800h CGA emulation
    B000h MDA emulation
    0000h no emulation
 08h  WORD  PG1600 graphics buffer segment
 0Ah  WORD  number of bytes between consecutive graphic rows
 0Ch  WORD  horizontal pixel size
 0Eh  WORD  vertical pixel size
 10h  WORD  horizontal dots per inch
 12h  WORD  vertical dots per inch
 14h  WORD  graphics buffer bits per pixel
 16h  WORD  monitor bits per pixel
--------N-214402-----------------------------
INT 21 - PC/TCP IPCUST.SYS - RESET CONFIGURATION DATA READ POINTER
  AX = 4402h
  BX = file handle referencing device "$IPCUST"
  CX, DS:DX ignored
Return: CF clear if successful
      AX destroyed
```

```
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Notes:  there are a total of 378h bytes of configuration data for IPCUST.SYS
    version 2.05.  If less than the entire data is read or written,
    the next read/write continues where the previous one ended; this
    call and AX=4403h both reset the location at which the next
    operation starts to zero
  v2.1+ uses a new configuration method, but allows the installation
    of IPCUST.SYS for backward compatibility with other software which
    must read the PC/TCP configuration
SeeAlso: AH=3Fh"IPCUST",AH=40h"IPCUST",AX=4403h"IPCUST"
--------N-214402-----------------------------
INT 21 - WORKGRP.SYS - GET API ENTRY POINT
  AX = 4402h
  BX = file handle for device "NET$HLP$"
  CX = 0008h
  DS:DX -> buffer for entry point record (see #01485)
Return: CF clear if successful
      AX = number of bytes actually read
  CF set on error
      AX = error code
Program: WORKGRP.SYS is the portion of Microsoft's Workgroup Connection which
    permits communication with PCs running Windows for Workgroups or
    LAN Manager
SeeAlso: AH=3Fh"WORKGRP.SYS"

Format of WORKGRP.SYS entry point record:
Offset  Size  Description (Table 01485)
 00h  WORD  3633h  \ signature???
 02h  WORD  EF6Fh  /
 04h  DWORD address of entry point (see #01486)
Note: first four bytes of buffer must be 6Fh E9h 33h 36h on entry when using
    IOCTL rather than READ to get the entry point record

(Table 01486)
Call WORKGRP entry point with:
  STACK:  WORD  function number (0000h-0009h)
Return: STACK unchanged
SeeAlso: #01487,#01488,#01489,#01490,#01491,#01492,#01493,#01494

(Table 01487)
```

```
Call WORKGRP function 00h with:
  STACK:  WORD  0000h (function "get ???")
Return: DX:AX -> data table

(Table 01488)
Call WORKGRP function 01h with:
  STACK:  WORD  0001h (function "hook ???")
Return: STACK:  DWORD pointer to ???
    WORD  0001h (function number)

(Table 01489)
Call WORKGRP function 02h with:
  STACK:  WORD  0002h (function "unhook ???")
  ???
Return: ???

(Table 01490)
Call WORKGRP function 03h with:
  STACK:  WORD  0003h (function "reenable printer port")
    WORD  LPT port number
Return: ???

(Table 01491)
Call WORKGRP function 04h with:
  STACK:  WORD  0004h (function "disable printer port")
    WORD  LPT port number
Return: ???

(Table 01492)
Call WORKGRP function 05h with:
  STACK:  WORD  0005h (function "???")
    ???
Return: ???

(Table 01493)
Call WORKGRP function 06h with:
  STACK:  WORD  0006h (function "???")
Return: STACK unchanged
  AX = 0000h
  DX = 0000h
```

```
(Table 01494)
Call WORKGRP functions 07h-09h with:
  STACK:  WORD  0007h-0009h (NOP functions)
Return: STACK unchanged
  AX = 0001h
  DX = 0000h
--------N-214402-----------------------------
INT 21 - 10NET v5.0 - 10BEUI.DOS - API
  AX = 4402h
  BX = file handle referencing device "10BEUI$"
  DS:DX -> parameter record (see #01495)
  CX ignored
Return: CF clear if successful
      AX destroyed
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
SeeAlso: AX=4402h"10MEMMGR",INT 6F/AH=00h"10NET"

Format of 10NET 10BEUI.DOS parameter record:
Offset  Size  Description (Table 01495)
 00h  WORD  000Ah (function number???)
 02h  WORD  ???
 04h  DWORD pointer to buffer for ???
 08h  4 BYTEs ???
 0Ch  WORD  transfer size
--------N-214402-----------------------------
INT 21 - 10NET v5.0 - 10MEMMGR.SYS - API
  AX = 4402h
  BX = file handle referencing device "MEMMGR0$"
  DS:DX -> 6-byte buffer for interface info (see #01496)
  CX ignored
Return: CF clear if successful
      AX destroyed
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
SeeAlso: AX=4402h"10BEUI",INT 6F/AH=00h"10NET"

Format of 10NET 10MEMMGR.SYS interface info:
Offset  Size  Description (Table 01496)
 00h  DWORD address of entry point (see #01497)
 04h  WORD  version (0500h for v5.00)
```

```
(Table 01497)
Call 10NET 10MEMMGR.SYS entry point with:
  AL = 01h ???
       BX = ???
       Return: CF clear if successful
         CF set on error
       AX = error code
  AL = 02h ???
       ???
  AL = 03h ???
       ???
  AL = 04h set/restore memory allocation strategy
       BX = subfunction
     0000h set strategy
     0001h restore strategy
       Return: CF clear if successful
         CF set on error (if function disabled)
         various registers destroyed
  AL = other
       Return: CF set
         AX = 0000h
         BL = 01h
--------V-214402-----------------------------
INT 21 - Compaq AG1024.SYS - RGDI - GET DRIVER LOCATION
  AX = 4402h
  BX = file handle for device "$$$$RGDI"
  CX = 0006h (size of returned data)
  DS:DX -> location record (see #01498)
Return: CF clear if successful
       buffer filled
  CF set on error
       AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Program: AG1024.SYS is a device driver for the Advanced Graphics 1024 adapter
SeeAlso: AX=4403h"RGDI"

Format of Compaq AG1024.SYS location record:
Offset  Size  Description (Table 01498)
 00h  WORD  signature 55AAh
 02h  WORD  segment of ???
 04h  WORD  segment of device driver's code
```

```
--------N-214402-----------------------------
INT 21 - FTPSOFT.DOS v3.1 - GET ???
  AX = 4402h
  BX = file handle for device "FTPSOFT$"
  CX = size of buffer
  DS:DX -> buffer for data (see #01499)
Return: CF clear if successful
      buffer filled
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Program: FTPSOFT.DOS is a device driver for Protocol Manager support from
     FTP Software, Inc.
SeeAlso: AH=3Fh"PC/TCP",AX=4402h"NDIS"

Format of FTPSOFT.DOS data:
Offset  Size  Description (Table 01499)
 00h  WORD  (call) BA98h (if different, no data returned)
 02h  DWORD -> NDIS common characteristics table
      (see #01435 at AX=4402h"NDIS")
 06h  DWORD (call) -> new dispatch table (see #01437 at AX=4402h"NDIS")
 0Ah  DWORD -> 28-byte buffer for ??? data
 0Eh  DWORD ???
 12h  DWORD -> FAR function to reset dispatch jump table to defaults
 16h  BYTE  ???
Note: the addresses in the new dispatch table are copied into an internal
    jump table which may be reset by calling the function pointed at by
    offset 12h
--------n-214402-----------------------------
INT 21 U - PenDOS PENDEV.SYS - GET ENTRY POINTS
  AX = 4402h
  BX = file handle for device "$$PENDOS" or "$$PD_REG"
  CX = size of buffer (4 for $$PENDOS and a 4,8,12, or 16 for $$PD_REG)
  DS:DX -> buffer for entry point record (see #01500)
Return: CF clear if successful
      buffer filled
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Program: A limited version of PenDOS by Communication Intelligence Corporation,
    which provides pen capability to keyboard-based programs, is bundled
    with IBM DOS 6.1
SeeAlso: AX=4403h"PENDEV.SYS"
```

```
Format of PENDEV.SYS entry point record:
Offset  Size  Description (Table 01500)
 00h  DWORD -> array of jumps
 04h  WORD  offset of function to retrieve entry point (see #01501)
 06h  2 BYTEs signature "Pe"
 08h  WORD  offset of function to set entry point (see #01502)
 0Ah  2 BYTEs signature "nD"
 0Ch  WORD  offset of function to clear entry point (see #01503)
 0Eh  WORD  signature "OS"

(Table 01501)
Call PENDEV.SYS function to retrieve entry point with:
  AX = index of entry point (0-9)
Return: CF clear if successful
      DX:AX -> desired entry point
  CF set on error (AX out of range)

(Table 01502)
Call PENDEV.SYS function to set entry point with:
  AX = index of entry point (0-9)
  DX:SI -> new handler
Return: CF clear if successful
  CF set on error (AX out of range)

(Table 01503)
Call PENDEV.SYS function to clear entry point with:
  AX = index of entry point (0-9)
Return: CF clear if successful
  CF set on error (AX out of range)
Note: resets the jump at the specified entry point to its default target,
    which simply returns
--------N-214402-----------------------------
INT 21 U - LAN Manager - TCPDRV.DOS - API
  AX = 4402h
  BX = file handle referencing device "TCPDRV$"
  CX = 0019h
  DS:DX -> buffer containing request block (see #01504)
Return: CF clear if successful
      buffer filled
  CF set on error
```

```
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Program: TCPDRV.DOS is the low-level device driver supporting LAN Manager's
    TCP/IP protocol stack

Format of TCPDRV.DOS request block:
Offset  Size  Description (Table 01504)
 00h  BYTE  (call) function number
    00h initialize ???
    06h get ???
    07h get ???
 01h  BYTE  (call) 00h
    (ret) error code if error, unchanged if successful
 02h  WORD  signature 4354h ('CT')
---function 00h---
 04h  DWORD (call) pointer to ??? FAR function
    function is called with ES:BX -> device driver request used to
      invoke this function
 08h  4 BYTEs ???
 0Ch  DWORD (call) pointer to ??? record, WORD at offset 22h is read
 10h  DWORD (ret) -> ??? buffer if 0000h:0000h on call
---function 06h---
 04h  4 BYTEs ???
 08h  DWORD (ret) pointer to ???
---function 07h---
 04h  DWORD (ret) pointer to ??? record
--------y-214402-----------------------------
INT 21 U - PC Tools 9 CPRLOW.EXE - GET CODE AND DATA ADDRESSES
  AX = 4402h
  BX = file handle referencing device "RECLOWLD"
  DS:DX -> buffer for address list (see #01505)
  CX ignored
Return: CF clear if successful
      buffer filled
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)

Format of CPRLOW address list:
Offset  Size  Description (Table 01505)
 00h  WORD  segment of CPRLOW code
 02h  WORD  offset in code segment of ??? entry point
    (switches into protected mode)
```

```
 04h  WORD  offset in code segment of jump array (see #01506)
 06h  WORD  segment of copy of interrupt vector table at CPRLOW load time
Note: neither the entry point nor the jump array is valid until after a
    CPR /LOAD, because CPR.EXE installs the code into CPRLOW at runtime.

Format of CPRLOW jump array:
Offset  Size  Description (Table 01506)
 00h  3 BYTEs initialize CPRLOW interrupt hooks
 03h  3 BYTEs reset timers and enable CPR (hotkey enable)
 06h  3 BYTEs disable CPR (hotkey disable)
 09h  3 BYTEs clear ??? flag, hotkey disable, and ???
 0Ch  3 BYTEs initialize delay loop counter (destroys AX,BX,CX,DX)
 0Fh  3 BYTEs disable CPR completely (commandline /DISABLE)
 12h  3 BYTEs enable ??? if CPR enabled by both cmdline and hotkey
 15h  3 BYTEs enable CPR (commandline /ENABLE)
--------s-214402-----------------------------
INT 21 U - Creative Technology CTMMSYS.SYS v1.00.01 - API
  AX = 4402h
  BX = file handle for device "CTMMSYS$"
  CX = 0004h (size of data)
  DS:DX -> buffer for entry point (see #01507)
Return: CF clear if successful
      buffer updated
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Program: CTMMSYS.SYS is the Creative DOS Multimedia Architecture Integration
    driver
SeeAlso: AX=4402h"CTSB2",INT 80/BX=0000h"SBFM"

Format of CTMMSYS.SYS entry point record:
Offset  Size  Description (Table 01507)
 00h  DWORD (call) signature 4D6D7443h (ASCII "CtmM")
    (ret) pointer to CTMMSYS entry point (see #01508)
SeeAlso: #01509

(Table 01508)
Call CTMMSYS.SYS entry point with:
  AX = ???
  STACK:  WORD  ???
    WORD  ???
    WORD  ??? (0001h,0002h,0005h,0006h)
```

```
    WORD  ???
    DWORD -> ???
    WORD  ???
    WORD  ???
Return: DX:AX = ??? or error code
    0000h:000Bh invalid value for ???
    0000h:000Fh API call already in progress
SeeAlso: #01507
--------s-214402-----------------------------
INT 21 U - Creative Technology CTSB2.SYS v1.01.01 - API
  AX = 4402h
  BX = file handle for device "CTSOUND0"
  CX = 0004h (size of data)
  DS:DX -> buffer for entry point (see #01509)
Return: CF clear if successful
      buffer updated
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Program: CTMMSYS.SYS is the Creative DOS Multimedia Architecture Integration
    driver
SeeAlso: AX=4402h"CTMMSYS",INT 80/BX=0000h"SBFM"

Format of CTSB2.SYS entry point record:
Offset  Size  Description (Table 01509)
 00h  DWORD (call) signature 4D6D7443h (ASCII "CtmM")
    (ret) pointer to CTSB2 entry point (see #01510)
SeeAlso: #01507

(Table 01510)
Call CTSB2.SYS entry point with:
  AX = ???
  STACK:  DWORD -> ???
    WORD  function number
      (0100h,0200h,0300h,0400h,0500h,0600h,0701h)
    DWORD -> ???
    WORD  ???
    WORD  ???
Return: DX:AX = ??? or error code
    0000h:0004h invalid subfunction???
    0000h:000Bh invalid value for ???
    0000h:000Fh API call already in progress
```

```
SeeAlso: #01509
--------m-214402-----------------------------
INT 21 U - Novell DOS 7+ EMM386.EXE - GET STATE RECORD
  AX = 4402h
  BX = file handle for device "EMMXXXX0" or "EMMQXXX0"
  CX = 003Eh (size of state record)
  DS:DX -> buffer for state record (see #01511)
Return: CF clear if successful
      buffer filled (see #03603 at INT 67/AH=3Fh)
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Note: This function is called by the Novell DOS 7+ MEM utility.

Format of Novell DOS 7 - DR-DOS 7.03 EMM386.EXE state record:
Offset  Size  Description (Table 01511)
 00h  WORD  signature EDC0h
 02h 12 BYTEs (ret) bitmap for include range
 0Eh 12 BYTEs (ret) bitmap for auto range
 1Ah 12 BYTEs (ret) bitmap for BIOS range
 26h 12 BYTEs (ret) bitmap for ROM range
 32h 12 BYTEs (ret) bitmap for map range
Note: each bitmap covers the upper-memory address range A000h-FFFFh, where
    offset X bit B within the bitmap indicates the state of the 4K page
    starting at segment A000h + (8X + B) * 100h
--------m-214402SF00-------------------------
INT 21 U - Memory Managers - GET API ENTRY POINT
  AX = 4402h subfn 00h
  BX = file handle for device "EMMXXXX0"
  CX = 0006h (size of buffer in bytes)
  DS:DX -> buffer for API entry point record (see #01512)
    first byte must be 00h on entry
Return: CF clear if successful
      buffer filled (see #03603 at INT 67/AH=3Fh)
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Notes:  this function is supported by Microsoft EMM386.EXE v4.45+ and
    CEMM v5.10+, and is intended for use by MS Windows as it starts up
  if no other program has hooked INT 67, an alternate installation
    check for CEMM is testing for the string
    "COMPAQ EXPANDED MEMORY MANAGER 386" at offset 14h in the INT 67
    handler's segment; if present, the word at offset 12h contains the
```

```
    offset of the API entry point
SeeAlso: AX=4402h/SF=01h,AX=4402h/SF=02h,AX=4402h"EMM386",INT 67/AH=3Fh

Format of memory manager API entry point record:
Offset  Size  Description (Table 01512)
 00h  WORD  ??? (0022h for CEMM 5.11, 0025h for MS EMM386 v4.45)
 02h  DWORD manager's private API entry point
    (see #01513,#03666 at INT 67/AX=FFA5h)

(Table 01513)
Call CEMM v5.10+ entry point with:
  AH = 00h get memory manager's state
       Return: AH = state
       bit 0: turned OFF
       bit 1: AUTO mode enabled
  AH = 01h set memory manager's state
       AL = new state (00h ON, 01h OFF, 02h AUTO)
       Return: CF clear if successful
         CF set on error
  AH = 02h Weitek coprocessor support
       AL = subfunction
    00h get Weitek support state
         Return: AL = status
         bit 0: Weitek coprocessor is present
         bit 1: Weitek support is enabled
    01h turn on Weitek support
    02h turn off Weitek support
      Return: CF clear if successful
        CF set on error
      AH = error code (01h invalid subfunc, 02h no Weitek)
  AH = 05h get statistics
       ???
  AH > 06h
       Return: CF set
         AH = 01h (invalid function)
Notes:  AH=03h,04h,06h are NOPs which return CF clear, presumably for backwards
    compatibility with earlier versions of CEMM
  in v5.11, AH=05h merely prints an error message (using INT 21/AH=09h)
    stating that a different version of CEMM is installed and it is
    therefore not possible to display the statistics
--------m-214402SF01-------------------------
```

```
INT 21 U - Memory Managers - GET EMM IMPORT STRUCTURE ADDRESS
  AX = 4402h subfn 01h
  BX = file handle for device "EMMXXXX0"
  CX = 0006h (size of buffer in bytes)
  DS:DX -> buffer for EMM import structure record (see #01514)
    first byte must be 01h on entry
Return: CF clear if successful
      buffer filled (see also #03603 at INT 67/AH=3Fh)
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Notes:  this function is supported by Microsoft EMM386.EXE v4.45+,
    QEMM-386 v6+, and CEMM v5.10+, and is intended for use by MS Windows
    as it starts up
  for QEMM-386, this call always returns an error if Windows3 support
    has been disabled with the NW3 switch
SeeAlso: AX=4402h/SF=00h,AX=4402h"EMM386",INT 2F/AX=D201h/BX=5145h
SeeAlso: INT 67/AH=3Fh

Format of EMM import structure record:
Offset  Size  Description (Table 01514)
 00h  DWORD physical address of EMM import structure (see #01515)
 04h  BYTE  EMM import structure major version
 05h  BYTE  EMM import structure minor version
Note: vession 1.00 contains only EMS information (Windows 3.0+)
  version 1.10 contains UMB/XMS/HMA/EMS information (Windows 3.1)
  version 1.11 is version 1.10 plus memory manager maker/product name
SeeAlso: #03643

Format of Global EMM Import record:
Offset  Size  Description (Table 01515)
 00h  BYTE  bit flags
    bit 2: ???
    bit 3: free EMM386 virtual HMA only if hma_page_table_paddr!=0
    bit 4: no UMB???
 01h  BYTE  reserved (0)
 02h  WORD  size of structure in bytes
 04h  WORD  structure version
 06h  DWORD reserved
 0Ah 384 BYTEs  64 EMS frame status records (see #01516), one per 16K of
      real-mode 1M address space
18Ah  BYTE  ??? (must be at least 3*number_of_EMS_frames+4)
```

```
18Bh  BYTE  number of UMB frame descriptors following
18Ch 4N DWORDs  UMB frame descriptors
    each is 4 DWORDs giving physical page numbers for the four
      4K pages of a 16K EMS frame (00000000h if non-UMB page)
var BYTE  number of EMS handle info records following
    16N BYTEs EMS handle info records (see #01518)
---version 1.10+ ---
  DWORD realmode INT 67 vector (used by Windows to set breakpoints)
  DWORD physical address of HMA page table values
  BYTE  number of free page entries following
     2N DWORDs  free page entries
    each is:
        DWORD physical page number
        DWORD number of consecutive physical pages
  BYTE  number of XMS handle info records following
    00h if memory manager does not emulate XMS or has real mode
      XMS code which can execute in the Windows environment
    12N BYTEs XMS handle info records (see #01519)
  BYTE  number of free UMB info records following
     2N WORDs free UMB info records
    each is:
        WORD  real mode start segment
        WORD  size in paragraphs
---version 1.11---
     20 BYTEs blank-padded maker name
     20 BYTEs blank-padded product name

Format of EMS frame status record:
Offset  Size  Description (Table 01516)
 00h  BYTE  frame type (see #01517)
 01h  BYTE  owner handle (00h/FFh = none) from frame including UMB
    index to UMB frame descriptors
 02h  WORD  logical page for frame, 7FFFh if none, FFFFh if non-EMS frame
 04h  BYTE  EMS physical page number (FFh for non-EMS = don't care???)
 05h  BYTE  flags for non-EMS frames (00h for EMS frame)
    bits 0,1 for first 4K, bits 2,3 for second 4K, etc:
      10: direct mapping (linear address = physical address)
      01: UMB mapping

Bitfields for EMS frame type:
Bit(s)  Description (Table 01517)
```

```
 0  EMS frame
 1  (if EMS frame) in standard 64K page frame
 2  first 4K of frame is UMB
 3  second 4K of frame is UMB
 4  third 4K of frame is UMB
 5  last 4K of frame is UMB

Format of EMS handle info record:
Offset  Size  Description (Table 01518)
 00h  BYTE  handle number (00h = system handle)
 01h  BYTE  flags
    bit 0: normal handle rather than system handle
    bit 2: ??? (set by some EMS managers)
 02h  8 BYTEs EMS handle's name
 0Ah  WORD  number of 16K pages for handle
 0Ch  DWORD physical address of page table entries forming page map
Note: all values should be zero for the system handle if no large frame
    support is present

Format of XMS handle info record:
Offset  Size  Description (Table 01519)
 00h  WORD  handle
 02h  WORD  flags
    bit 0: handle usable by Windows
      (already in use when Windows started if clear)
    bit 1: reserved (0)
 04h  DWORD size in KB (may be zero, used only if flags bit 0 set)
 08h  DWORD physical address (only if flags bit 0 set)
--------m-214402SF02-------------------------
INT 21 U - Memory Managers - GET MEMORY MANAGER VERSION
  AX = 4402h subfn 02h
  BX = file handle for device "EMMXXXX0"
  CX = 0002h (size of buffer in bytes)
  DS:DX -> buffer for memory manager version (see #01520)
    first byte must be 02h on entry
Return: CF clear if successful
      buffer filled
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Note: this function is supported by Microsoft EMM386.EXE v4.45+ and
    CEMM v5.10+, and is intended for use by MS Windows as it starts up
```

```
SeeAlso: AX=4402h/SF=00h,AX=4402h"EMM386",INT 67/AH=3Fh

Format of memory manager version:
Offset  Size  Description (Table 01520)
 00h  BYTE  major version
 01h  BYTE  minor version (binary)
--------m-214402-----------------------------
INT 21 U - Microsoft EMM386.EXE v4.45 - GET MEMORY MANAGER INFORMATION
  AX = 4402h
  BX = file handle for device "EMMXXXX0"
  CX = size of buffer in bytes (varies, see #01521)
  DS:DX -> buffer for returned data (see #01521)
    first byte must be set on entry to indicate desired data
Return: CF clear if successful
      buffer filled
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Notes:  an error is returned if the number of bytes to be read does not match
    the number of bytes returned for the specified data item
  this function is part of the interface which allows MS Windows to
    cooperate with memory managers
SeeAlso: AX=4402h/SF=00h,AX=4402h/SF=01h,AX=4402h/SF=02h,INT 67/AX=FFA5h

Format of EMM386.EXE data buffer:
Offset  Size  Description (Table 01521)
 00h  BYTE  (call) function
    03h get ???
    04h get ???
---function 03h---
 00h  WORD  ???
 02h  WORD  ???
---function 04h---
 00h  WORD  segment of UMB containing EMM386 code/data
 02h  WORD  number of paragraphs of EMM386 code/data in UMB
 04h  WORD  ???
----------214402-----------------------------
INT 21 U - IFSHLP.SYS - GET ENTRY POINT
  AX = 4402h
  BX = file handle for device "IFS$HLP$"
  CX = 0008h (size of buffer in bytes)
  DS:DX -> buffer for entry point record (see #01415 at AH=3Fh"IFSHLP")
```

```
Return: CF clear if successful
      buffer filled
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
SeeAlso: AH=3Fh"IFSHLP"
--------d-214402----------------------------
INT 21 - ATA Software Programming Interface (ATASPI) - INTERFACE
  AX = 4402h
  BX = file handle for device "$ATAMGR$"
  CX = 0004h (size of buffer in bytes)
  DS:DX -> buffer for result (see #90000), set to zeros before call
Return: CF clear if successful
      AX = 0004h
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Notes:  ATASPI is an API to control IDE, EIDE, and ATAPI devices, proposed by
    Future Domain; there's at least one driver in existence which
    conforms to this specification: ATASPI16.SYS by Future Domain
  (see #90002) on the stack and the caller must clean up the stack
  Future Domain ATASPI16.SYS v2.2 performs ATASPI calls using INT 4Fh
    interface instead of this INT 21/4402h interface and itself supports
    both interfaces
SeeAlso: AX=4402h"ASPI",INT 4F/AX=0081h,INT 4F/AX=0082h,PORT 01F0h-01F7h"HDC1"

Format of ATASPI IOCTL result:
Offset  Size  Description (Table 90000)
 00h  DWORD ATASPI entry point address

(Table 90001)
Call ATASPI entry point with:
  STACK:  DWORD -> ATA Request Block (ARB) (see #90003)
Notes:  caller must clean up the stack
  the large-model C declaration is
    void (*entry)(ARB_Struct *);
SeeAlso: #90000

(Table 90002)
Values for ATASPI command code:
 00h  ATA controller inquiry
 01h  get ATA device type
 02h  execute ATA I/O
```

```
 03h  abort ATA request
 04h  reset ATA device
 05h  set ATA controller parameter
 06h  get ATA disk drive information
SeeAlso: #90003

Format of ATA Request Block (ARB):
Offset  Size  Description (Table 90003)
 00h  BYTE  command code (see #90002)
 01h  BYTE  (ret) status (see #90004)
 02h  BYTE  ATA controller number
 03h  BYTE  request flags (see #90005)
 04h  DWORD reserved (0)
---request 00h---
 08h  BYTE  (ret) total number of ATA controllers
    valid only if ATA controller number = 0FFh
 09h  BYTE  reserved (0)
 0Ah 16 BYTEs (ret) ATA manager ID
 1Ah 16 BYTEs (ret) ATA controller ID
 2Ah 16 BYTEs (ret) controller unique parameters (see #90010)
---request 01h---
 08h  BYTE  device ID (00h = master, 01h = slave)
 09h  BYTE  reserved (0)
 0Ah  BYTE  (ret) peripheral device type (see #90007)
---request 02h---
 08h  BYTE  device ID
 09h  BYTE  reserved (0)
 0Ah  DWORD data  transfer length
    (ret) residual byte length
 0Eh  BYTE  sense allocation length (N)
 0Fh  DWORD data  buffer pointer
 13h  DWORD reserved (0)
 17h  BYTE  ACB length (M)
    for Task File requests must be set to 07h, for ATAPI Packet
      requests must match the number of bytes in the packet
 18h  BYTE  (ret) ATA controller status (see #90006)
 19h  BYTE  (ret) device status
    value of the error register, 00h if no error (see #P0512)
 1Ah  DWORD post routine  address (see #90009)
 1Eh  WORD  data transfer block size (in bytes)
    number of data bytes to transfer per hardware interrupt for Task
```

```
      File requests; number of data bytes host prefers to transfer
      per hardware interrupt for ATAPI Packet commands
 20h 32 BYTEs reserved for ATASPI workspace
 40h  M BYTEs ATA/ATAPI command block (ACB)
    contains AT Task File Structure (see #90008) if bit 2 of the
      request flags is set, ATAPI packet (see #03236,#03237,#03238)
      if bit 2 is cleared
40h+M N BYTEs sense allocation area
---request 03h---
 08h  DWORD address of ARB to be aborted
---request 04h---
 08h  BYTE  device ID
 09h 15 BYTEs reserved (0)
 18h  BYTE  (ret) ATA controller status (see #90006)
 19h  BYTE  (ret) device status
 1Ah  DWORD post routine address (see #90009)
 1Eh 34 BYTEs reserved for ATASPI workspace
---request 05h---
 08h 16 BYTEs controller unique parameters (vendor unique)
---request 06h---
 08h  BYTE  device ID
 09h  BYTE  reserved (0)
 0Ah  BYTE  (ret) drive flags (see #01461)
 0Bh  BYTE  (ret) INT 13h drive
 0Ch  BYTE  (ret) preferred head   translation
 0Dh  BYTE  (ret) preferred sector translation
 0Eh 10 BYTEs reserved (0)

(Table 90004)
Values for ATASPI request status:
 00h  request in progress
 01h  request completed without error
 02h  request aborted by host
 04h  request completed with error
 80h  invalid request
 81h  invalid ATA controller number
 82h  ATA device not installed
 83h  ATA controller/device busy
Note: if ATASPI ever returns 83h as the status, it is the responsibility of
    the driver/applications to re-send the request at a later time
SeeAlso: #90003
```

```
Bitfields for request flags:
Bit(s)  Description (Table 90005)
 7  reserved (0)
 6  "ByteXfer"
  =0 use word transfer mode
  =1 use byte transfer mode
 5  (ATAPI device) DSC unavailable action (DUA)
  =0 queue the request and service it when DSC bit is set
  =1 return with status 83h
 4-3  direction
  00 direction determined by device
  01 data in
  10 data out
  11 no data transfer
 2  request type
  =0 ATAPI Packet Command
  =1 AT Task File Structure
 1  reserved (0)
 0  "Post"
  =0 disable posting
  =1 enable  posting
SeeAlso: #90003

(Table 90006)
Values for ATA controller status:
 00h  no error
 11h  device not present
 12h  data overrun/underrun
SeeAlso: #90003

(Table 90007)
Values for peripheral device type:
 00h  direct-access device (e.g. magnetic disk)
 01h  tape device (QIC-121 SCSI Architectural Model)
 02h-04h reserved
 05h  CD-ROM device
 06h  reserved
 07h  optical memory device (e.g. some optical disks)
 08h-0Bh reserved
 0Ch  tape device (Cost Sensitive Architectural Model)
```

```
 0Dh-1Eh reserved
 1Fh  unknown or no device type
 80h  non-ATAPI device
SeeAlso: #90003

Format of AT Task File Structure:
Offset  Size  Description (Table 90008)
 00h  features register
 01h  sector count register
 02h  cylinder LSB register
 04h  cylinder MSB register
 05h  device/head  register (see #P0513)
 06h  command register (see #P0515)
SeeAlso: #90005

(Table 90009)
Values ATASPI post function is called with:
  STACK:  DWORD -> ARB (see #90003) which completed
  interrupts disabled
Return: EBP, EBX, ESI, EDI must be preserved
  interrupts disabled
Notes:  the post function may issue any ATASPI function except an abort; it
    should complete as quickly as possible
  the large-model C declaration is
    void (*post)(ARB_Struct *);
SeeAlso: #90003

Format of Future Domain controller unique parameters:
Offset  Size  Description (Table 90010)
 00h  WORD  controller features
 02h  WORD  controller main I/O port
 04h  WORD  controller alternate I/O port
 06h  BYTE  controller IRQ
 07h  BYTE  ??? (00h or 01h)
 08h  BYTE  ??? (00h or 01h)
 09h  7 BYTEs reserved (0)
SeeAlso: #90003
----------214402-----------------------------
INT 21 - DRFAT32.SYS device driver - IOCTL INPUT
  AX = 4402h
  BX = file handle referencing character device for DRFAT32.SYS driver
```

```
      (e.g. "FAT32XXX" in the default configuration)
  CX = number of bytes to read
  DS:DX -> control block (see #04108)
Return: CF clear if successful
      AX = number of bytes actually read
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Notes:   the data returned depends on the first byte of the control block
     the remainder of the control block is filled by the driver
   one DRFAT32.SYS device driver supports up to 8 subunits (that is up
     to 8 FAT32 partitions). By default, DRFAT32.SYS logs in all FAT32
     partitions it finds, however, by using command line options /C (CHS),
     /L (LBA), /H (Hidden) and /P:1..4 (Primary), /E:1..255 (Logical
     Drive), and /U:min[,max] (Unit 128..255) it can be directed to
     attach only to one or a specific range of partitions. Multiple
     DRFAT32.SYS drivers can be loaded in a system when using the
     /D:name option. To assign them to the redirector, the /D:name option
     must be repeated for each of the DRFAT32.SYS drivers in the
     DRFAT32.EXE command line.
SeeAlso: AX=4402h"CD-ROM",INT 2F/AX=0802h

Format of DRFAT32 control block:
Offset  Size  Description (Table 04108)
 00h  BYTE  data being requested
    00h device driver header address
    09h media change status
    48h address of DRFAT32 geometry table
    else error
---function 00h---
 01h  DWORD device driver header address (see also AH=52h,#01646)
---function 09h---
 01h  BYTE  media change status
    00h don't know
    01h media unchanged
    FFh media has been changed
---DRFAT32 function 48h---
 01h  DWORD address of DRFAT32 geometry table (see #04109)

Format of DRFAT32 Geometry Table:
Offset  Size  Description (Table 04109)
 00h 8*59 BYTEs eight DRFAT32 partition data tables (see #04110)
```

```
 1D8h BYTE  DRFAT32 access flags (see #04111)
 1D9h BYTE  DRFAT32 status flags (see #04112)
Note: This is the table layout used by DRFAT32.SYS 1.00 which supports up
    to 8 sub-units. The layout of this table and its records may change,
    so the version indicator in the device driver's signature ("FAT320")
    and the size of the public data structure should be checked first.
SeeAlso: #04108

Format of DRFAT32 partition data table:
Offset  Size  Description (Table 04110)
 00h  BYTE  size of following public data structure (9)
  --- public data (needed by the DRFAT32.EXE redirector) ---
 01h  WORD  bytes per sector
 03h  BYTE  sectors per cluster
 04h  WORD  reserved sectors
 06h  DWORD root directory start cluster
  --- private data ---
 0Ah  DWORD sectors per track
 0Eh  DWORD root directory sector
 12h  DWORD partition start sector
 16h  DWORD highest partition sector allowed
    (only if range checking enabled)
 1Ah  DWORD absolute cluster start sector
    (cluster start sector+partition start sector)
 1Eh  DWORD total sectors per track (number of heads * sectors per track)
 22h  BYTE  drive unit (default 80h)
 23h  BYTE  temp buffer: CHS sectors to write (if verify enabled)
 24h  BYTE  INT 13h extension version
  --- CHS data ---
 25h  WORD  cylinder
 27h  WORD  head
 29h  WORD  sector
  --- LBA data ---
 2Bh 16 BYTEs disk address packet (see #00272 at INT 13/AH=42h)
SeeAlso: #04109

Bitfields for DRFAT32 access flags:
Bit(s)  Description (Table 04111)
 7  using LBA addressing instead of CHS
 6  using INT 13h extensions (see also INT 13/AH=48h)
 5  multi-sector access allowed
```

```
 4  force media change indication on removable drives
 3  force verify after every write (/W)
 2  more than 1024 cylinders, less than 64 heads (/A)
 1  client-side asynchronous buffering allowed (/B), normally =0
 0  read-only access (/R)
SeeAlso: #04109,#04112

Bitfields for DRFAT32 status flags:
Bit(s)  Description (Table 04112)
 7-2  reserved (0)
 1  Save Guard enabled (blocks any further writes after a fault)
 0  volume may have dirty sectors (set after write operations)
SeeAlso: #04109,#04111
--------D-214403-----------------------------
INT 21 - DOS 2+ - IOCTL - WRITE TO CHARACTER DEVICE CONTROL CHANNEL
  AX = 4403h
  BX = file handle referencing character device
  CX = number of bytes to write
  DS:DX -> data to write
Return: CF clear if successful
      AX = number of bytes actually written
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Notes:  format of data is driver-specific
    (also see separate entries below for some specific cases)
  if the file handle refers to "4DOSSTAK", the 4DOS (v2.x-3.03)
    KEYSTACK.SYS driver will push the specified characters on the
    keyboard stack; similarly for "NDOSSTAK", the NDOS KEYSTACK.SYS
    driver will push the characters onto the keyboard stack
  this function was not supported by Digital Research's DOS Plus
SeeAlso: AX=4400h,AX=4402h"DOS 2+",AX=4405h"DOS",INT 2F/AX=122Bh
SeeAlso: INT 2F/AX=D44Dh,INT 2F/AX=D44Fh
--------c-214403-----------------------------
INT 21 - SMARTDRV.SYS v3.x only - IOCTL - CACHE CONTROL
  AX = 4403h
  BX = handle for device "SMARTAAR"
  CX = number of bytes to write
  DS:DX -> SMARTDRV control block (see #01523)
Return: CF clear if successful
      AX = number of bytes actually written
    0000h if control block too small for given command
```

```
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Note: NCache2 (from the Norton Utilities v8.0) attempts to support this
    interface, but does not do so correctly, often hanging the system;
    one should use the SmartDrive v4.x or NCache private interfaces
    (see INT 2F/AX=4A10h/BX=0000h,INT 2F/AX=FE00h/DI=4E55h)
SeeAlso: AX=4402h"SMARTDRV",INT 2F/AX=4A10h/BX=0000h

(Table 01522)
Values for SMARTDRV function code:
 00h  flush cache
 01h  flush and discard cache
 02h  disable caching (flushes and discards cache first)
 03h  enable caching
 04h  control write caching
 05h  set flushing tick count
 06h  lock cache contents
 07h  unlock cache contents
 08h  set flush-on-reboot flag
 09h  unused
 0Ah  control full-track caching
 0Bh  reduce cache size
 0Ch  increase cache size
 0Dh  set INT 13 chain address

Format of SMARTDRV control block:
Offset  Size  Description (Table 01523)
 00h  BYTE  function code (see #01522)
---functions 00h-03h,06h,07h---
 no additional fields
---function 04h---
 01h  BYTE  write caching control action
    00h turn off write-through
    01h turn on write-through
    02h turn off write buffering (also flushes cache)
    03h turn on write buffering (also flushes cache)
---function 05h---
 01h  WORD  number of timer ticks between cache flushes
---function 08h---
 01h  BYTE  new flush-on-reboot flag (00h off, 01h on)
---function 0Ah---
```

```
 01h  BYTE  full-track writes are
    00h not cached
    01h cached
---functions 0Bh,0Ch---
 01h  WORD  number of 16K pages by which to increase/reduce cache size
---function 0Dh---
 01h  DWORD new address to which to chain on INT 13
Note: the previous address is not preserved
--------d-214403-----------------------------
INT 21 - CD-ROM device driver - IOCTL OUTPUT
  AX = 4403h
  BX = file handle referencing character device for CD-ROM driver
  CX = number of bytes to write
  DS:DX -> control block (see #01524)
Return: CF clear if successful
      AX = number of bytes actually written
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Notes:  some device drivers support several subunits (that is more than one
    drive) but it is not possible to distinguish between them with this
    function; use INT 2F/AX=1510h or INT 2F/AX=0802h instead
  under Windows95, the "mscd$$$$" device cannot be opened so you cannot
    obtain the handle required by this function: use INT 2F/AX=1510h
    or INT 2F/AX=0802h instead
SeeAlso: AX=4402h"CD-ROM",INT 2F/AX=0802h,INT 2F/AX=1510h

Format of CR-ROM control block:
Offset  Size  Description (Table 01524)
 00h  BYTE  function code
    00h eject disk
    01h lock/unlock door
    02h reset drive
    03h control audio channel
    04h write device control string
    05h close tray
---functions 00h,02h,05h---
 no further fields
---function 01h---
 01h  BYTE  lock function
    00h unlock door
    01h lock door
```

```
---function 03h---
 01h  BYTE  input channel (0-3) for output channel 0
 02h  BYTE  volume for output channel 0
 03h  BYTE  input channel (0-3) for output channel 1
 04h  BYTE  volume for output channel 1
 05h  BYTE  input channel (0-3) for output channel 2
 06h  BYTE  volume for output channel 2
 07h  BYTE  input channel (0-3) for output channel 3
 08h  BYTE  volume for output channel 3
Note: output channels 0 and 1 are left and right, 2 and 3 are left prime and
    right prime; a volume of 00h is off
---function 04h---
 01h  N BYTEs bytes to send directly to the CD-ROM drive without
      interpretation
--------D-214403-----------------------------
INT 21 U - MS-DOS 7.0+ - CONFIG$ device - SET??? CONFIGURATION INFORMATION
  AX = 4403h
  BX = file handle for character device "CONFIG$"
  CX = number of bytes to write
  DS:DX -> buffer containing ???
Return: CF clear if successful
      AX = number of bytes actually written
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Note: IOCTL Output to CONFIG$ is only available while IO.SYS is booting
    the system; once CONFIG.SYS has been processed, this call always
    succeeds, ignoring any input
SeeAlso: AX=4403h"CONFIG$"
--------d-214403-----------------------------
INT 21 - Brian Antoine Seagate ST-01 SCSI.SYS - IOCTL - EXECUTE COMMANDS
  AX = 4403h
  BX = handle for device "SCSITAPE"
  CX = number of bytes to write
  DS:DX -> SCSITAPE control block (see #01525)
Return: CF clear if successful
      AX = number of bytes actually written
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
SeeAlso: AX=4405h"ST-01",INT 78/AH=10h

Format of SCSITAPE control block:
```

```
Offset  Size  Description (Table 01525)
 00h  WORD  command type
    'F' Format (argument 1 = interleave, argument 2 = format type)
    'E' Erase
    'R' Rewind
    'L' Load
    'N' No Load
    'S' Space (argument 1 = count, argument 2 = type)
    'M' File Mark (argument 1 = count)
    'A' Reassign
 02h  WORD  argument 1
 04h  WORD  argument 2
 06h  WORD  segment of command buffer
 08h  WORD  offset of command buffer
 0Ah  WORD  length of command buffer
--------E-214403-----------------------------
INT 21 U - AI Architects - OS/x86??? - API
  AX = 4403h
  BX = handle for device "AIA_OS"
  CX = number of bytes to write (ignored)
  DS:DX -> 12-byte buffer (see #01526), first byte is command:
       81h installation check
       82h get API entry point
       84h uninstall
Return: CF clear if successful
       AX = number of bytes actually written
       DS:DX buffer filled
  CF set on error
       AX = error code (01h,05h,06h,0Ch,0Dh) (see #01680 at AH=59h)
Notes:  these functions are only available if the DOS extender was loaded as a
    device driver in CONFIG.SYS
  called by TKERNEL (a licensed version of AI Architects/Ergo OS/x86)
SeeAlso: INT 2F/AX=FBA1h/BX=0081h,INT 2F/AX=FBA1h/BX=0082h
Index:  installation check;OS/x86|entry point;OS/x86|uninstall;OS/x86

Format of buffer on return:
Offset  Size  Description (Table 01526)
 00h  4 BYTEs signature "IABH"
---if function 81h---
 (no additional fields)
---if function 82h---
```

```
 04h  DWORD pointer to API entry point (see INT 2F/AX=FBA1h/BX=0082h)
---if function 84h---
 04h  WORD  success indicator
 06h  WORD  segment of ???
 08h  WORD  segment of ??? memory block to free if nonzero
 0Ah  WORD  segment of ??? memory block to free if nonzero
--------V-214403-----------------------------
INT 21 - PGS1600.DEV - IOCTL - SET CONFIGURATION???
  AX = 4403h
  BX = file handle for device "PGS1600$"
  CX = 0018h (size of buffer)
  DS:DX -> configuration buffer (see #01484 at AX=4402h"PGS1600")
Return: CF clear if successful
      AX = number of bytes actually written
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Program: PGS1600.DEV is a device driver for the Cornerstone Technology PG1600
    display adapter, which provides a 1600x1200 monochrome display as
    well as one of two emulations, MDA or CGA.
SeeAlso: AX=4402h"PGS1600"
--------N-214403-----------------------------
INT 21 - PC/TCP IPCUST.SYS - RESET CONFIGURATION DATA READ POINTER
  AX = 4403h
  BX = file handle referencing device "$IPCUST"
  CX, DS:DX ignored
Return: CF clear if successful
      AX destroyed
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Notes:  there are a total of 378h bytes of configuration data for IPCUST.SYS
    version 2.05.  If less than the entire data is read or written,
    the next read/write continues where the previous one ended; this
    call and AX=4402h both reset the location at which the next
    operation starts to zero
  v2.1+ uses a new configuration method, but allows the installation
    of IPCUST.SYS for backward compatibility with other software which
    must read the PC/TCP configuration
SeeAlso: AH=3Fh"IPCUST",AH=40h"IPCUST",AX=4402h"IPCUST"
--------V-214403-----------------------------
INT 21 - Compaq AG1024.SYS - CPQ_MGES - IOCTL OUTPUT
  AX = 4403h
```

```
  BX = file handle referencing device "CPQ_MGES"
  DS:DX -> request packet (see #01527)
  CX ignored
Return: CF clear if successful
      AX destroyed
      data buffer filled (if applicable)
      first word of request packet set to number of bytes of data
        available (amount returned is smaller of this and requested
        amount)
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Program: AG1024.SYS is a device driver for the Advanced Graphics 1024 adapter
SeeAlso: AX=4403h"RGDI"

Format of Compaq AG1024.SYS request packet:
Offset  Size  Description (Table 01527)
 00h  WORD  function
    0000h get ??? (26h bytes)
    0001h get ??? (36h bytes)
    0002h set ??? (same as returned by function 0001h)
    0003h get ??? (6 bytes)
    0004h get ???
    0005h get ???
    0006h get ??? (10h bytes)
    0007h set ??? (same as returned by function 0006h)
---functions 00h-03h,06h,07h---
 02h  WORD  size of data buffer
 04h  DWORD -> buffer for function's data
 ---functions 04h,05h---
 02h  WORD  ???
 04h  WORD  size of data buffer
 06h  DWORD -> buffer to receive data
--------V-214403-----------------------------
INT 21 - Compaq AG1024.SYS - RGDI - IOCTL OUTPUT
  AX = 4403h
  BX = file handle referencing device "$$$$RGDI"
  DS:DX -> request packet (see #01528)
  CX ignored
Return: CF clear if successful
      AX destroyed
      data buffer filled (if applicable)
```

```
      first word of request packet set to number of bytes of data
        available (amount returned is smaller of this and requested
        amount)
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Program: AG1024.SYS is a device driver for the Advanced Graphics 1024 adapter
SeeAlso: AX=4402h"RGDI",AX=4403h"CPQ_MGES"

Format of Compaq AG1024.SYS request packet:
Offset  Size  Description (Table 01528)
 00h  WORD  function
    0000h get entry points
    0001h get ???
 02h  DWORD address of buffer for returned data
--------m-214403SF01-------------------------
INT 21 U - Qualitas 386MAX v6.01+ - TURN 386MAX OFF
  AX = 4403h subfn 01h
  BX = handle for device "386MAX$$"
  DS:DX -> BYTE 01h
  CX ignored
Return: DS:DX -> BYTE status (00h = successful)
  CF clear if successful
      AX destroyed
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Note: this function will fail if any EMS or UMBs are in use
SeeAlso: AX=4403h/SF=02h,AX=4403h/SF=03h,AX=4403h/SF=05h
--------m-214403SF02-------------------------
INT 21 U - Qualitas 386MAX v6.01+ - TURN 386MAX ON
  AX = 4403h subfn 02h
  BX = handle for device "386MAX$$"
  DS:DX -> BYTE 02h
  CX ignored
Return: DS:DX -> BYTE status (00h = successful)
  CF clear if successful
      AX destroyed
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
SeeAlso: AX=4403h/SF=01h,AX=4403h/SF=03h,AX=4403h/SF=05h
--------m-214403SF03-------------------------
INT 21 U - Qualitas 386MAX v6.01+ - SET STATE
```

```
  AX = 4403h subfn 03h
  BX = handle for device "386MAX$$"
  CX = number of bytes to copy (up to size of state buffer)
  DS:DX -> BYTE 03h followed by state buffer
       (see #01467 at AX=4402h"386MAX")
Return: CF clear if successful
       AX = number of bytes actually written
  CF set on error
       AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Note: the first byte of the buffer must be either 01h, 02h, or 03h
    (specifying the version of the state record) and the buffer must
    contain CX bytes AFTER the initial byte
SeeAlso: AX=4402h"386MAX"
--------m-214403SF05-------------------------
INT 21 U - Qualitas 386MAX v7.00+ - LIMIT AUTOMATIC ACTIVATION TO STD EMS CALLS
  AX = 4403h subfn 05h
  BX = handle for device "386MAX$$"
  DS:DX -> BYTE 05h
  CX ignored
Return: CF clear if successful
       AX destroyed
  CF set on error
       AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Desc: specifies that 386MAX will only shift from AUTO to ON mode on standard
    EMS calls INT 67/AH=40h-5Dh
SeeAlso: AX=4403h/SF=02h,AX=4403h/SF=03h,AX=4403h/SF=06h
--------m-214403SF06-------------------------
INT 21 U - Qualitas 386MAX v7.00+ - ALLOW AUTOMATIC ACTIVATION ON ANY INT 67
  AX = 4403h subfn 06h
  BX = handle for device "386MAX$$"
  DS:DX -> BYTE 06h
  CX ignored
Return: CF clear if successful
       AX destroyed
  CF set on error
       AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Desc: specified that 386MAX should shift from AUTO to ON mode on any INT 67
    call other than INT 67/AH=3Fh
SeeAlso: AX=4403h/SF=01h,AX=4403h/SF=03h,AX=4403h/SF=05h
--------n-214403----------------------------
INT 21 U - PenDOS PENDEV.SYS - ???
```

```
  AX = 4403h
  BX = file handle for device "$$PENDOS" or "$$PD_REG"
  CX = size of buffer
  DS:DX -> buffer containing ???
Return: CF clear if successful
      buffer filled
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Program: A limited version of PenDOS by Communication Intelligence Corporation,
    which provides pen capability to keyboard-based programs, is bundled
    with IBM DOS 6.1
Note: this call sets the WORD at offset 1Ah into the device driver request
    header used to call the driver to 0000h.
SeeAlso: AX=4402h"PENDEV.SYS"
----------214403-----------------------------
INT 21 - DR DOS $IDLE$ - IOCTL - DYNAMIC IDLE DETECTION CONTROL
  AX = 4403h
  BX = file handle referencing character device $IDLE$
  CX = number of bytes to write (0004h)
  DS:DX -> $IDLE$ IOCTL structure (see #04113)
Return: CF clear if successful
      AX = number of bytes actually written
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Notes:  When the DR DOS IBMBIO.COM detects the presence of the $IDLE$ device
    during startup, it will retrieve the address of the idle state
    data area via INT 21/AX=4458h and pass it along to the $IDLE$ device
    driver by calling this function.

Format of DR DOS $IDLE$ IOCTL structure:
Offset  Size  Description (Table 04113)
 00h  DWORD -> idle state data area (see Table !!! at INT 21/AX=4458h)
--------D-214404-----------------------------
INT 21 - DOS 2+ - IOCTL - READ FROM BLOCK DEVICE CONTROL CHANNEL
  AX = 4404h
  BL = drive number (00h = default, 01h = A:, etc.)
  CX = number of bytes to read
  DS:DX -> buffer
Return: CF clear if successful
      AX = number of bytes actually read
  CF set on error
```

```
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Notes:  format of data is driver-specific
  this function was not supported by Digital Research's DOS Plus
SeeAlso: AX=4402h"DOS 2+",AX=4405h"DOS",INT 2F/AX=122Bh
--------k-214404-----------------------------
INT 21 - Stacker - GET DEVICE DRIVER ADDRESS AND SET VOLUME NUMBER
  AX = 4404h
  BL = drive number (00h = default, 01h = A:, etc.)
  CX = 0004h
  DS:DX -> DWORD buffer to receive device driver address
Return: buffer filled with pointer into Stacker device driver (see #02550):
    driver + 19h if Stacker Anywhere controls this drive
    driver + 1Ah if Stacker controls this drive
    unchanged else
Notes:  in addition to returning the address of the Stacker device driver,
    this call also sets the volume number at offset 58h in the device
    driver (see #02550 at INT 25/AX=CDCDh)
  Stacker Anywhere does not link its built-in device driver into
    the standard device driver chain, but it can be found via CDS/DPB
  this call can be used as an installation check for all versions of
    Stacker and Stacker Anywhere to avoid the INT 25 call
SeeAlso: AX=4408h,AX=440Eh,AH=52h,INT 25/AX=CDCDh
--------k-214404-----------------------------
INT 21 - Stacker - GET STACVOL FILE SECTORS
  AX = 4404h
  BL = drive number (0 is current drive)
  CX = byte count (i.e., 200h = 1 sector)
  DS:DX -> buffer (see #01529)
Return: Data Buffer contains the number of sectors requested from the
    STACVOL physical file for the drive specified.

Format of Stacker buffer:
Offset  Size  Description (Table 01529)
 00h  WORD  01CDh
 02h  WORD  sector count
 04h  DWORD number of starting sector
 08h  DWORD far pointer to Data Buffer
--------k-214404-----------------------------
INT 21 - DUBLDISK.SYS v2.6 - GET INFO
  AX = 4404h
  BL = drive number of DoubleDisk drive (00h = default, 01h = A:, etc.)
```

```
  CX = number of bytes (000Ah-0014h, call ignored otherwise)
  DS:DX -> data record (see #01530)
Return: CF clear if successful
      AX = number of bytes read
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Program: DUBLDISK.SYS is the device driver portion of DoubleDisk, a disk
    expander by Vertisoft Systems, Inc.
InstallCheck: for v2.6, scan memory for the signature "FAT 2.6  byte:",
    which is immediately followed by a data table (see #01531)
BUG:  VOPT (a disk optimizer by Golden Bow Software) assumes that any driver
    which returns the "correct" number of bytes when the first word of
    the buffer for the data record contains the signature value 4444h is
    DoubleDisk; a workaround is for the non-DoubleDisk driver to return
    no data if the signature is present
SeeAlso: AX=440Dh"DOS 3.2+"
Index:  installation check;DUBLDISK.SYS

Format of DUBLDISK data record:
Offset  Size  Description (Table 01530)
 00h  WORD  (call) signature 4444h
 02h  BYTE  (call) function
      00h ???
      01h ???
---function 00h---
 02h  BYTE  (ret) ???
 03h  BYTE  (ret) ???
---function 01h---
 02h  WORD  (ret) 4444h
 04h  WORD  allocation unit size???
 06h  WORD  ???
 08h  WORD  ???
 0Ah  BYTE  ???

Format of DUBLDISK signature data table:
Offset  Size  Description (Table 01531)
 00h  5 BYTEs ???
 05h  BYTE  first drive number
 06h  BYTE  number of drives
 07h  ???
--------k-214404-----------------------------
```

```
INT 21 - DBLSPACE.BIN - IOCTL - FLUSH OR INVALIDATE INTERNAL CACHES
  AX = 4404h
  BL = drive number (00h = default, 01h = A:, etc)
  CX = 000Ah (size of DSPACKET structure)
  DS:DX -> DSPACKET structure (see #01532)
Return: CF clear if IOCTL successful -- check DSPACKET for actual status
      AX = number of bytes actually transferred
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
SeeAlso: AX=4405h"DBLSPACE",INT 2F/AX=4A11h/BX=0000h

Format of DoubleSpace DSPACKET structure:
Offset  Size  Description (Table 01532)
 00h  WORD  signature 444Dh ("DM")
 02h  BYTE  command code
    46h ('F') flush internal caches
    49h ('I') flush and invalidate internal caches
 03h  WORD  result code
    (ret) 4F4Bh ("OK") if successful, else unchanged
 05h  5 BYTEs padding
--------k-214404-----------------------------
INT 21 - DBLSPACE.BIN v6.2 - IOCTL - GET ??? FOR SPECIFIED DRIVE
  AX = 4404h
  BL = drive number (00h = default, 01h = A:, etc)
  CX = size of DSPACKET structure (ignored in DOS 6.2)
  DS:DX -> DSPACKET structure (see #01533)
Return: CF clear if IOCTL successful -- check DSPACKET for actual status
      AX = number of bytes actually transferred
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
SeeAlso: AX=4405h"DBLSPACE",INT 2F/AX=4A11h/BX=0000h

Format of DoubleSpace DSPACKET structure:
Offset  Size  Description (Table 01533)
 00h  WORD  signature 444Dh ("DM")
 02h  BYTE  command code
    53h ('S') get ??? for specified drive
 03h  WORD  result code
    (ret) 4F4Bh ("OK") if successful, else unchanged
 05h  DWORD (ret) pointer to 96-byte ??? data
 09h  DWORD (ret) pointer to ??? data
```

```
 0Dh  3 BYTEs reserved
--------k-214404-----------------------------
INT 21 U - DoubleTools v1.0 - GET ???
  AX = 4404h
  BL = drive number (00h = default, 01h = A:, etc)
  CX = 0006h
  DS:DX -> DoubleTools structure (see #01534)
Return: CF clear if IOCTL successful -- check DSPACKET for actual status
      AX = number of bytes actually transferred
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Note: the Transporter device driver has the signature 55h 61h 50h 74h
    ("DaPt") twelve bytes after it beginning

Format of DoubleTools structure:
Offset  Size  Description (Table 01534)
 00h  WORD  (call) signature 55h AAh
    (ret) signature 78h 70h ("xp")
 02h  WORD  (call) signature 52h 16h
    (ret) ???
 04h  WORD  (ret) segment of driver's DS (same as its PSP)
--------d-214404-----------------------------
INT 21 U - xDISK v3.31 - CONFIGURE
  AX = 4404h
  BL = drive number (00h = default, 01h = A:, etc)
  CX = 0047h (length of version string)
  DS:DX -> 79-byte buffer for version string and ???
  DS:0081h = commandline containing new switches for driver
Return: CF clear if successful
      AX = number of bytes actually transferred
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Program: xDISK is a shareware resizeable EMS RAMdisk by FM de Monasterio
SeeAlso: AX=4405h"xDISK",INT 2F/AH=DDh/BX=7844h"xDISK"
--------c-214404-----------------------------
INT 21 - COMBI-disk v1.13 - GET DATA RECORD
  AX = 4404h
  BL = drive number (00h = default, 01h = A:, etc)
  CX = 0032h (length of data packet)
  DS:DX -> buffer for data packet (see #01535)
Return: CF clear if successful
```

```
      AX = number of bytes actually transferred
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Program: COMBI-disk is a shareware combination cache and RAMdisk sharing a
    single pool of memory by Vadim V. Vlasov
InstallCheck: scan the valid drives for one which returns a correctly-sized
    data packet with the proper signature in the first field
SeeAlso: AX=4405h"COMBI"
Index:  installation check;COMBI-disk

Format of COMBI-disk data packet:
Offset  Size  Description (Table 01535)
 00h  6 BYTEs ASCIZ signature "COMBI"
 06h  WORD  version (high byte = major, low = decimal minor version)
 08h  BYTE  current options
 09h  BYTE  sectors per allocation block
 0Ah  WORD  maximum buffer in KB
 0Ch  WORD  current buffer in KB (less than max if XMS memory being lent)
 0Eh  WORD  total number of allocation blocks
 10h  WORD  current number of allocation blocks
 12h  WORD  number of blocks being used by RAM disk
 14h  WORD  number of blocks being used by cache or unused
 16h  WORD  number of dirty cache blocks
 18h  WORD  number of blocks which could not be written out due to errors
 1Ah  WORD  total number of read requests
 1Ch  WORD  total number of sectors read
 1Eh  WORD  number of BIOS read requests (cache misses)
 20h  WORD  number of sectors read via BIOS (cache misses)
 22h  WORD  total number of write requests
 24h  WORD  total number of sectors written
 26h  WORD  number of BIOS write requests
 28h  WORD  number of sectors written via BIOS
 2Ah  WORD  number of RAM disk read requests
 2Ch  WORD  number of sectors read from RAM disk
 2Eh  WORD  number of RAM disk write requests
 30h  WORD  number of sectors written to RAM disk
--------d-214404-----------------------------
INT 21 - SFS v1.00 - GET CONFIGURATION INFORMATION
  AX = 4404h
  BL = drive number (00h = default, 01h = A:, etc)
  CX = 001Ah (length of data packet)
```

```
  DS:DX -> buffer for data packet (see #01536)
Return: CF clear if successful
      AX = number of bytes actually transferred
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Program: SFS (Secure FileSystem) is a shareware encrypting disk driver by
    Peter C. Gutmann
SeeAlso: AX=4405h"SFS"

Format of SFS data packet:
Offset  Size  Description (Table 01536)
 00h  4 BYTEs signature string "SFS1"
 04h  WORD  SFS unit number (0-based)
 06h  WORD  drive on which the SFS volume is mounted
 08h  DWORD sector offset of logical volume from start of physical volume
    0 if logical volume = physical volume
 0Ch  WORD  flag: 00h = no disk mounted, 01h = disk mounted
 0Eh  WORD  flag: 00h read/write, 01h read-only
 10h  WORD  quick-unmount hotkey
    (high byte = shift state, low = scan code) (see #00006)
 12h  WORD  auto-unmount time in minutes, or 0000h if already expired, or
    FFFFh if not set
 14h  WORD  time in minutes remaining before auto-unmount
 16h  WORD  internal driver check code
    0000h no error
    0001h driver consistency check failed
    0002h unit consistency check failed
 18h  WORD  disk access mode
    0000h BIOS
    0001h IDE direct access
    0002h SCSI direct access
--------D-214405-----------------------------
INT 21 - DOS 2+ - IOCTL - WRITE TO BLOCK DEVICE CONTROL CHANNEL
  AX = 4405h
  BL = drive number (00h = default, 01h = A:, etc)
  CX = number of bytes to write
  DS:DX -> data to write
Return: CF clear if successful
      AX = number of bytes actually written
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
```

```
Notes:  format of data is driver-specific
  this function was not supported by Digital Research's DOS Plus
SeeAlso: AX=4403h"DOS",AX=4404h"DOS",INT 2F/AX=122Bh
--------d-214405-----------------------------
INT 21 - Brian Antoine Seagate ST-01 SCSI.SYS - IOCTL - EXECUTE COMMANDS
  AX = 4405h
  BX = drive number (00h = default, 01h = A:, etc)
  CX = number of bytes to write
  DS:DX -> SCSIDISK control block (see also #01525 at AX=4403h"ST-01")
Return: CF clear if successful
      AX = number of bytes actually written
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
SeeAlso: AX=4403h"ST-01"
--------k-214405-----------------------------
INT 21 U - DBLSPACE.BIN - IOCTL - FLUSH OR INVALIDATE INTERNAL CACHES
  AX = 4405h
  BL = drive number (00h = default, 01h = A:, etc)
  CX = 000Ah (size of DSPACKET structure)
  DS:DX -> DSPACKET structure (see #01537)
Return: CF clear if IOCTL successful -- check DSPACKET for actual status
      AX = number of bytes actually transferred
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Note: this call is identical to the documented AX=4404h
SeeAlso: AX=4404h"DBLSPACE",INT 2F/AX=4A11h/BX=0000h

Format of DoubleSpace DSPACKET structure:
Offset  Size  Description (Table 01537)
 00h  WORD  signature 444Dh ("DM")
 02h  BYTE  command code
    46h ('F') flush internal caches
    49h ('I') flush and invalidate internal caches
 03h  WORD  result code
    (ret) 4F4Bh ("OK") if successful, else unchanged
 05h  5 BYTEs padding
--------d-214405-----------------------------
INT 21 U - xDISK v3.31 - ???
  AX = 4405h
  BL = drive number (00h = default, 01h = A:, etc)
  CX = number of bytes to write
```

```
  DS:DX -> buffer containing version string
  ???
Return: CF clear if successful
      AX = number of bytes actually transferred
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Program: xDISK is a shareware resizeable EMS RAMdisk by FM de Monasterio
SeeAlso: AX=4404h"xDISK",INT 2F/AH=DDh/BX=7844h"xDISK"
--------d-214405-----------------------------
INT 21 - COMBI-disk v1.13 - CONTROL COMBI-disk
  AX = 4405h
  BL = drive number (00h = default, 01h = A:, etc) for RAM disk
  CX = number of bytes to write
  DS:DX -> buffer containing command packet (see #01539)
Return: CF clear if successful
      AX = number of bytes actually transferred
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
SeeAlso: AX=4404h"COMBI"

(Table 01538)
Values for COMBI-disk command code:
 80h  flush cache
 81h  change options byte
 82h  shrink memory
 83h  expand memory
 84h  get dirty block information
 85h  reset write errors
 86h  reset counters

Format of COMBI-disk command packet:
Offset  Size  Description (Table 01539)
 00h  WORD  version
 02h  BYTE  command code (see #01538)
---command code 80h---
 no additional fields
---command code 81h---
 03h  BYTE  new options byte (see #01540)
---command code 82h---
 03h  WORD  number of KB to release
---command code 83h---
```

```
 03h  WORD  number of KB to expand
---command code 84h---
 03h  DWORD -> buffer for block info (see #01541)
---command code 85h---
 03h  DWORD block ID
---command code 86h---
 03h  BYTE  which counters to reset
    bit 0: hard disk read counts
    bit 1: hard disk write counts
    bit 2: RAM disk read/write counts
Note: multiple commands may be placed in a single packet by stringing
    together as many command/argument pairs as desired

Bitfields for COMBI-disk options byte:
Bit(s)  Description (Table 01540)
 0  cache off
 1  cache frozen
 2  write caching enabled
 3  delayed writing disabled
 5  fix memory allocation (no XMS lending)
 6  no 'sector not found' error

Format of COMBI-disk block info:
Offset  Size  Description (Table 01541)
 00h  DWORD block ID
 04h  BYTE  bitmask of valid sectors in block
 05h  BYTE  bitmask of dirty sectors in block
 06h  BYTE  last error returned by BIOS
 07h  BYTE  number of errors
--------d-214405-----------------------------
INT 21 - SFS v1.00 - DRIVER CONTROL
  AX = 4405h
  BL = drive number (00h = default, 01h = A:, etc)
  CX = number of bytes to write
  DS:DX -> data to write (see #01543)
Return: CF clear if successful
      AX = number of bytes actually written
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Program: SFS (Secure FileSystem) is a shareware encrypting disk driver by
    Peter C. Gutmann
```

```
SeeAlso: AX=4404h"SFS"

(Table 01542)
Values for SFS control function:
 00h  "PACKET_SET_DISKINFO"  set disk parameters
 01h  "PACKET_SET_KEYINFO"   set keying information
 02h  "PACKET_SET_READONLY"  select read/write or read-only
 03h  "PACKET_SET_DRIVENO"   set drive number to mount
 04h  "PACKET_SET_MOUNTSTATUS" set mount status
 05h  "PACKET_SET_UNMOUNT"   set/clear quick-unmount hotkey
 06h  "PACKET_SET_TIMEOUT"   set/clear auto-unmount timeout

Format of SFS control data packet:
Offset  Size  Description (Table 01543)
 00h  WORD  signature 4330h ('C0')
 02h  WORD  function (see #01542)
---function 00h---
 04h  WORD  sector size in bytes
 06h  BYTE  sectors per cluster
 07h  WORD  number of boot sectors
 09h  BYTE  number of copies of FAT
 0Ah  WORD  size of root directory in entries
 0Ch  WORD  number of sectors on disk, 16-bit
 0Eh  BYTE  media descriptor byte
 0Fh  WORD  sectors per FAT
 11h  WORD  sectors per track
 13h  WORD  number of heads
 15h  DWORD number of hidden sectors
 19h  DWORD number of sectors on disk, 32-bit
---function 01h---
 04h 20 BYTEs master IV for encrypted disk
 18h 64 BYTEs NDC/SHS keying information
---function 02h---
 04h  WORD  read-only state: 00h read-only, 01h read/write
---function 03h---
 04h  WORD  drive number (see #01544)
 06h  DWORD sector offset of logical volume from start of physical volume
    0 if logical volume = physical volume
---function 04h---
 04h  WORD  mount status (00h unmounted, 01h mounted)
---function 05h---
```

```
 04h  WORD  hotkey (high byte = shift state, low byte = scan code or 00h)
    (see #00006)
    0000h to disable hotkey
---function 06h---
 04h  WORD  timeout in minutes before automatic unmount
    0000h to disable auto-unmount
Notes:  the data for function 00h corresponds to a DOS BPB (see AH=53h)
  functions 00h, 01h, and 03h automatically unmount the encrypted drive
  unmounting a drive with function 04h also destroys the encryption
    information in the driver and forces all dirty buffers to be flushed

Bitfields for SFS drive number:
Bit(s)  Description (Table 01544)
 15-12  drive access mode
  0000 BIOS access
  0001 direct IDE access
  0010 direct SCSI access
---if BIOS access---
 11-8 unused (0)
 7-0  BIOS drive number
---if IDE access---
 11-8 unused (0)
 7-0  IDE drive number
---if SCSI access---
 11-8 SCSI host number
 7-4  SCSI target ID
 3-0  SCSI logical unit number
----------214405-----------------------------
INT 21 U - SUPERSTOR - IOCTL - INSTALL CHECK
  AX = 4405h
  BL = drive number (00h = default, 01h = A:, etc)
  (BH = 0???)
  CX = 000Ch (size of SuperStor packet structure)
  DS:DX -> SuperStor packet structure (see #04114)
Return: CF clear if IOCTL successful -- check SuperStor packet for actual
        status
      AX = number of bytes actually transferred???
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Note: This function is called with CX = 0008h for command 06h by NWCACHE
    although the Request Packet can be larger.  WINSWAP.SYS calls it
```

```
    with 000Ch - also for command 06h.

Format of SuperStor packet:
Offset  Size  Description (Table 04114)
 00h  WORD  (call) product signature AA55h
    (ret) result code, 0000h if successful
 02h  BYTE  (call) SuperStor product ID (01h)
 03h  BYTE  (call) SuperStor command parameter:
    06h = get compressed drive structure / drive parameters
    07h = return the compressed length of a file
    08h = return real drive ID for swapped drive
    09h = validate the contents of the cluster
    0Bh = checks whether deleted cluster is free
    0Ch = reallocate previously deleted cluster
    0Dh = read absolute sector from the drive
    0Eh = write absolute sector to the drive
    0Fh = instruct the driver to rebuild tables
    10h = flush any cached Replacement Block Table (RBT) sectors
          to the drive
---command 06h---
 04h  DWORD (ret) pointer to SuperStor CVF's compressed unit structure
          (see #04115)
 08h  DWORD (ret) pointer to internal status structure
---command 07h---
 04h  DWORD (ret) size of file in bytes
 08h  WORD  (ret) first cluster number
---command 08h---
 04h  BYTE  (ret) swapped drive (0-based)
---commands 09h,0Bh,0Ch---
 04h  WORD  (call) cluster
---commands 0Dh,0Eh---
 04h  DWORD (call) first logical sector to transfer
 08h  WORD  (call) number of logical sectors to transfer
 0Ah  DWORD (call) pointer to buffer
---command 0Fh---
 04h  WORD  (call) temporary buffer area
---command 10h---
 no additional parameters

Format of SuperStor CVF compressed unit structure:
Offset  Size  Description (Table 04115)
```

```
 00h 31 BYTEs BIOS parameter block for compressed drive (see #01663)
 1Fh  5 BYTEs reserved for future expansion of the BPB
 24h 33 BYTEs   DOS 4-6 style drive parameter block for compressed drive
      (see #01395)
 45h  2 BYTEs reserved for future expansion of the DPB
 47h  WORD  first sector of relocation table (RBT)
 49h  WORD  first sector of file allocation table
 4Bh  WORD  first sector of the data area
 4Dh  BYTE  sec2byteshift
 4Eh  2 BYTEs ???
    Note: Reportedly these two bytes did not exist in ADDSTOR's
        specification, so the whole structure would have
        been documented differently.
 50h  WORD  sectors per cluster
 52h  BYTE  sector shift
 53h  WORD  compression ratio
 55h  DWORD number of physical sectors in the host file
 59h  WORD  OS version
 5Bh  BYTE  SuperStor level
 5Ch  BYTE  unknown???
 5Dh  BYTE  flags
    bits 7-1: ???
    bit 0: drive is mounted
 5Eh  BYTE  unknown???
 5Fh  DWORD pointer to CVF's underlaying host DCB
    this should be the physical DPB for this unit
        ->  +00h BYTE host unit DOS unit number
            +01h BYTE CVF's driver host subunit number
Note: As this SuperStor compressed unit structure's structure is located
    within the segment the SuperStor driver has occupied, the segment
    value of its address can be used as an installation check, since it
    contains the signature "ADDSTOR" at offset 20h in the SuperStor
    driver.  This is used by Novell DOS 7 WINSWAP.SYS to check for the
    presence of SuperStor.
SeeAlso: #04114
Index:  installation check;SuperStor
--------D-214406-----------------------------
INT 21 - DOS 2+ - IOCTL - GET INPUT STATUS
  AX = 4406h
  BX = file handle
Return: CF clear if successful
```

```
      AL = input status
    00h not ready (device) or at EOF (file)
    FFh ready
      AH may be destroyed (refer to note)
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Notes:  files may not register as being at EOF if positioned there by AH=42h
  under DOS 5.0, on a successful return, AH contains either the next
    character which will be read or 1Ah if at EOF; under a Windows95
    DOS box, AH seems to be either unchanged or 00h
  this function was not supported by Digital Research's DOS Plus
BUG:  the CLOCK$ device under DR DOS 3.41 always indicates 'Ready'; this was
    corrected in v5.0
SeeAlso: AX=4407h,INT 2F/AX=122Bh
--------D-214407-----------------------------
INT 21 - DOS 2+ - IOCTL - GET OUTPUT STATUS
  AX = 4407h
  BX = file handle
Return: CF clear if successful
      AL = input status
    00h not ready
    FFh ready
  CF set on error
      AX = error code (01h,05h,06h,0Dh) (see #01680 at AH=59h/BX=0000h)
Notes:  for DOS 2+, files are always ready for output, even if the disk is
    full or no media is in the drive
  this function was not supported by Digital Research's DOS Plus
SeeAlso: AX=4406h,INT 2F/AX=122Bh
--------D-214408-----------------------------
INT 21 - DOS 3.0+ - IOCTL - CHECK IF BLOCK DEVICE REMOVABLE
  AX = 4408h
  BL = drive number (00h = default, 01h = A:, etc)
Return: CF clear if successful
      AX = media type (0000h removable, 0001h fixed)
  CF set on error
      AX = error code (01h,0Fh) (see #01680 at AH=59h/BX=0000h)
Notes:  in addition to the normal operation, if Stacker is installed, this
    call also sets the volume number at offset 58h in the Stacker
    device driver (except under DR DOS 3.41-5.0, which do not pass
    through this call to the driver; use AX=440Eh instead)
    (see AX=4404h"Stacker",AX=440Eh,INT 25/AX=CDCDh)
```

```
  this function was not supported by Digital Research's DOS Plus
SeeAlso: AX=4400h,AX=4404h"Stacker",AX=4409h,INT 2F/AX=122Bh
--------D-214409-----------------------------
INT 21 - DOS 3.1+ - IOCTL - CHECK IF BLOCK DEVICE REMOTE
  AX = 4409h
  BL = drive number (00h = default, 01h = A:, etc)
Return: CF clear if successful
      DX = device attribute word
    bit 15: drive is SUBSTituted
    bit 13: (DR DOS 3.41/5.0 local drives only) always set
      media ID needed
    bit 12: drive is remote
    bit  9: direct I/O not allowed
  CF set on error
      AX = error code (01h,0Fh,15h) (see #01680 at AH=59h/BX=0000h)
Notes:  on local drives, DX bits not listed above are the attribute word from
    the device driver header (see #01646 at AH=52h); for remote drives,
    the other bits appear to be undefined for MS-DOS versions prior to
    5.0 (they are all cleared in DOS 5+)
  checking whether DX=0800h on return appears to be a fairly reliable
    method for detecting Microsoft's RAMDRIVE, though not for other
    ramdisks (there appears to be no simple yet foolproof method for
    detecting ramdisks, although the presence of only a single copy of
    the FAT and only a single head on non-removable devices is a fairly
    good indicator); for Windows95, RAMDRIVE returns DX=4800h
  for non-existent remote drives, this function sometimes returns
    AX=0015h (drive not ready) instead of AX=000Fh (invalid drive) on
    the first call; a subsequent call will return the correct error
    code
  this function was not supported by Digital Research's DOS Plus
SeeAlso: AX=4400h,AX=4408h,AX=440Ah,INT 2F/AX=122Bh
--------D-21440A-----------------------------
INT 21 - DOS 3.1+ - IOCTL - CHECK IF HANDLE IS REMOTE
  AX = 440Ah
  BX = handle
Return: CF clear if successful
      DX = attribute word (as stored in SFT)
    bit 15: set if remote
    bit 14: date/time not set on close
  CF set on error
      AX = error code (01h,06h) (see #01680 at AH=59h/BX=0000h)
```

```
Notes:  if file is remote, Novell Advanced NetWare 2.0 returns the number of
    the file server on which the handle is located in CX
  DR DOS 3.41 and 5.0 clear all bits of DX except bit 15; Novell DOS 7
    fully supports this function
  this function was not supported by Digital Research's DOS Plus
SeeAlso: AX=4400h,AX=4409h,AH=52h,INT 2F/AX=122Bh
--------D-21440B-----------------------------
INT 21 - DOS 3.1+ - IOCTL - SET SHARING RETRY COUNT
  AX = 440Bh
  CX = pause between retries (default 1)
  DX = number of retries (default 3)
Return: CF clear if successful
  CF set on error
      AX = error code (01h) (see #01680 at AH=59h/BX=0000h)
Notes:  delay is dependent on processor speed (value in CX specifies number of
    64K-iteration empty loops to execute)
  if DX=0000h on entry, the retry count is left unchanged
  this function was not supported by Digital Research's DOS Plus
SeeAlso: AH=52h,INT 2F/AX=1224h,INT 2F/AX=122Bh
--------D-21440C-----------------------------
INT 21 - DOS 3.2+ - IOCTL - GENERIC CHARACTER DEVICE REQUEST
  AX = 440Ch
  BX = device handle
  CH = category code (see #01545)
  CL = function number (see #01546)
  DS:DX -> parameter block (see #01547,#01548,#01549,#01550,#01551,#01552)
  SI = parameter to pass to driver (European MS-DOS 4.0, OS/2 comp box)
  DI = parameter to pass to driver (European MS-DOS 4.0, OS/2 comp box)
Return: CF set on error
      AX = error code (see #01680 at AH=59h/BX=0000h)
  CF clear if successful
      DS:DX -> iteration count if CL=65h
      SI = returned value (European MS-DOS 4.0, OS/2 comp box)
      DI = returned value (European MS-DOS 4.0, OS/2 comp box)
Note: DR DOS 3.41 and 5.0 return error code 16h on CL=45h,65h if the device
    does not support a retry counter
SeeAlso: AX=440Dh"DOS 3.2+",INT 2F/AX=0802h,INT 2F/AX=122Bh,INT 2F/AX=14FFh
SeeAlso: INT 2F/AX=1A01h

(Table 01545)
Values for IOCTL category code:
```

```
 00h  unknown (DOS 3.3+)
 01h  COMn: (DOS 3.3+)
 02h  reserved for terminal control
 03h  CON (DOS 3.3+)
 04h  reserved for keyboard control
 05h  LPTn:
 07h  mouse control (European MS-DOS 4.0)
 08h  reserved for disk control -- same as for block devices
  (see INT 21/AX=440Dh)
 9Eh  Media Access Control driver (STARLITE)
 00h-7Fh reserved for Microsoft
 80h-FFh reserved for OEM/user-defined
SeeAlso: #01558

(Table 01546)
Values for generic character IOCTL function:
 00h  MAC driver Bind (STARLITE) (see #01547)
 40h  DOS 3??? only: was reserved for set screen mode (see #04116)
 45h  set iteration (retry) count (see #01548)
 4Ah  select code page (see #01549)
 4Ch  start code-page preparation (see #01550)
 4Dh  end code-page preparation (see #01551)
 5Fh  set display information (DOS 4.0+) (see #01552)
 60h  DOS 3??? only: was reserved for get screen mode (see #04116)
 65h  get iteration (retry) count
 6Ah  query selected code page (see #01549)
 6Bh  query codepage prepare list (DOS 4.0+) (see #01553)
 7Fh  get display information (DOS 4.0+) (see #01552)
Note: bit assignments for function code:
      bit 7: set to ignore if unsupported, clear to return error
      bit 6: set if passed to driver, clear if intercepted by DOS
      bit 5: set if queries data from device, clear if sends command
    (by convention, if a function both queries and sends data,
      bit 5 should be clear)
      bits 4-0: subfunction

Format of parameter block for function 00h:
Offset  Size  Description (Table 01547)
 00h  8 BYTEs ASCIZ signature "STARMAC"
 08h  WORD  version
 0Ah  WORD  flags
```

```
    bit 0: media requires connect or listen request before use
    bit 1: network is a LAN (broadcast/multicast supported)
    bit 2: point-to-point network
 0Ch  WORD  handle for use with MAC driver's private interface (filled in
    by MAC driver)
 0Eh  WORD  context
 10h  WORD  approximate speed in KB/sec (filled in by MAC driver)
 12h  WORD  approximate cost in cents per hour (filled in by MAC driver)
 14h  WORD  maximum packet size in bytes (filled in by MAC driver)
 16h  WORD  addressing format (filled in by MAC driver)
    0000h general addressing
    0001h Ethernet addressing
    0002h Token Ring addressing
    0003h Token Bus addressing
 18h  DWORD Send entry point (filled in by MAC driver)
 1Ch  DWORD RegisterEventHandler entry point (filled in by MAC driver)
 20h  DWORD SetPacketFilter entry point (filled in by MAC driver)
 24h  DWORD UnBind entry point (filled in by MAC driver)

Format of parameter block for DOS 3??? screen mode functions 40h and 60h:
Offset  Size  Description (Table 04116)
 00h  WORD  length of folowing data (0009h)
 02h  BYTE  mode type
 03h  WORD  number of colors
 05h  WORD  width
 07h  WORD  height
Note: these functions appear never to have been released to the public
SeeAlso: #01546

Format of parameter block for function 45h:
Offset  Size  Description (Table 01548)
 00h  WORD  number of times output is attempted before driver assumes
      device is busy

Format of parameter block for functions 4Ah and 6Ah:
Offset  Size  Description (Table 01549)
 00h  WORD  length of data
 02h  WORD  code page ID (see #01757 at INT 21/AX=6602h)
 04h 2N BYTEs DCBS (double byte character set) lead byte range
      start/end for each of N ranges (DOS 4.0)
  WORD  0000h  end of data (DOS 4.0)
```

```
Format of parameter block for function 4Ch:
Offset  Size  Description (Table 01550)
 00h  WORD  flags
    DISPLAY.SYS = 0000h
    PRINTER.SYS bit 0 clear to prepare downloaded font, set to
      prepare cartridge selection
 02h  WORD  length of remainder of parameter block
 04h  WORD  number of code pages following
 06h  N WORDs code page 1,...,N

Format of parameter block for function 4Dh:
Offset  Size  Description (Table 01551)
 00h  WORD  length of data
 02h  WORD  code page ID (see #01757 at INT 21/AX=6602h)

Format of parameter block for functions 5Fh and 7Fh:
Offset  Size  Description (Table 01552)
 00h  BYTE  level (0 for DOS 4.x-6.0)
 01h  BYTE  reserved (0)
 02h  WORD  length of following data (14)
 04h  WORD  control flags
    bit 0 set for blink, clear for intensity
    bits 1-15 reserved
 06h  BYTE  mode type (1=text, 2=graphics)
 07h  BYTE  reserved (0)
 08h  WORD  colors
    0000h = monochrome
    else N bits per pixel
 0Ah  WORD  pixel columns
 0Ch  WORD  pixel rows
 0Eh  WORD  character columns
 10h  WORD  character rows

Format of parameter block for function 6Bh:
Offset  Size  Description (Table 01553)
 00h  WORD  length of following data
 02h  WORD  number of hardware code pages
 04h  N WORDs hardware code pages 1,...,N
  WORD  number of prepared code pages
      N WORDs prepared code pages 1,...,N
```

```
--------d-21440C-----------------------------
INT 21 - Greg Shenaut ASPITAPE.SYS - INTERFACE
  AX = 440Ch
  BX = device handle
  CH = category code
       07h tape (ASPITAPE.SYS)
  CL = function
       01h "mtop" - perform tape operation
       02h "mtget" - get tape status
       03h ignore end-of-tape errors
       04h enable end-of-tape errors
  DS:DX -> parameter block (see #01554,#01555)
Return: CF set on error
       AX = error code (see #01680 at AH=59h/BX=0000h)
  CF clear if successful
       DS:DX -> data block
Notes:  This device driver is a simple DOS interface to the Adaptec Advanced
    SCSI Programming Interface (ASPI).  It provides the following device
    names as access to the SCSI tape, 'RMTx' (rewind on close) and
    'NRMTx' (NO rewind on close) where x can go from 0 to 3.  There may
    also be the following names 'MTx' and 'NMTx' which default to 1024
    byte blocks.  The names may also have a '$' appended to try and make
    them unique from file names of 'RMT0' etc.
  once opend these devices must be put into RAW mode
SeeAlso: AX=4402h"ASPI"

Format of ASPITAPE.SYS mtop parameter block:
Offset  Size  Description (Table 01554)
 00h  WORD  operation code
    00h "MTWEOF" - write an end-of-file record
    01h "MTFSF" - forward space file
    02h "MTBSF" - backward space file
    03h "MTFSR" - forward space record
    04h "MTBSR" - backward space record
    05h "MTREW" - rewind
    06h "MTOFFL" - rewind and unload
    07h "MTNOP" - perform TEST UNIT READY
 02h  DWORD repetition count

Format of ASPITAPE.SYS mtget parameter block:
Offset  Size  Description (Table 01555)
```

```
 00h  BYTE  ASPI host ID
 01h  BYTE  SCSI target ID
 02h  BYTE  SCSI logical unit number
 03h  BYTE  device parameters
    bit 0: drive must use fixed-block read and write
    bit 7: drive is an ASPI device
 04h  BYTE  current device state (see #01556)
 05h  BYTE  unit number within driver
 06h  WORD  fixed block blocksize
 08h  BYTE  last SCSI status
 09h  BYTE  last SCSI sense key
 0Ah  WORD  last SCSI opcode (packed) (see #01557)
 0Ch  WORD  residual bytes from SCSI opcode

Bitfields for ASPITAPE.SYS current device state:
Bit(s)  Description (Table 01556)
 0  device currently opened in buffered mode
 1  drive currently opened in nonbuffered mode
 2  rewind drive on last close
 3  drive has been written on
 4  drive has been read from
 5  next read will return 0 bytes
 6  EOM will resemble EOF
 7  drive may be busy rewinding

Bitfields for SCSI opcode:
Bit(s)  Description (Table 01557)
 0-7  SCSI operation (SCSI packet byte 0)
 8-10 SCSI flags (SCSI packet byte 1)
 11-12  ASPI "Direction Bits" (ASPI SRB byte 3)
--------D-21440D-----------------------------
INT 21 - DOS 3.2+ - IOCTL - GENERIC BLOCK DEVICE REQUEST
  AX = 440Dh
  BL = drive number (00h=default,01h=A:,etc)
  CH = category code (see #01558)
  CL = minor code (function) (see #01559)
  DS:DX -> (DOS) parameter block (see #01560,#01562,#01563,#01564,#01565)
  SI:DI -> (OS/2 comp box) parameter block (see #01566,#01568,#01569,#01572)
Return: CF set on error
      AX = error code (01h,02h,etc.) (see #01680 at AH=59h/BX=0000h)
  CF clear if successful
```

```
      DS:DX -> data block if CL=60h or CL=61h
Notes:  DOS 4.01 seems to ignore the high byte of the number of directory
    entries in the BPB for diskettes.
  functions 46h and 66h undocumented in DOS 4.x, documented for DOS 5+
  the DUBLDISK.SYS v2.6 driver only supports minor codes 60h and 67h
  DR DOS 3.41-6.0 only support minor codes 40h-42h and 60h-62h; all
    other minor codes return error code 16h
  some PCMCIA calls reportedly appear to be dangerous for MS-DOS versions
    prior to 5.0
  minor code 60h normally produces no I/O except with AutoMount=1 for
    DBLSPACE/DRVSPACE
SeeAlso: AX=440Ch,AX=440Dh/CX=084Ah,AX=440Dh/CX=0871h,AH=69h,INT 2F/AX=0802h
SeeAlso: INT 2F/AX=122Bh

(Table 01558)
Values for block device IOCTL category code:
 08h  disk drive
 48h  FAT32 disk drive
 00h-7Fh reserved for Microsoft
 80h-FFh reserved for OEM/user-defined

(Table 01559)
Values for generic block IOCTL minor code:
 00h  (OS/2)  \ used to lock/unlock a drive
 01h  (OS/2)  /
 40h  set device parameters (see #01560)
 41h  write logical device track (see #01562)
 42h  format and verify logical device track (see #01563)
 46h  (DOS 4.0+) set volume serial number (see #01565,AH=69h)
 47h  (DOS 4.0+) set access flag (see #01566)
 48h  (Enh. Disk Drive Spec) set media lock state (see #01567,INT 13/AH=45h)
 49h  (Enh. Disk Drive Spec) eject media in drive (see INT 13/AH=49h)
  no parameter block required
 4Ah  (MS-DOS 7.0) lock logical volume (see AX=440Dh/CX=084Ah)
 4Bh  (MS-DOS 7.0) lock physical volume (see AX=440Dh/CX=084Bh)
 50h  (PCMCIA) attribute memory write
 51h  (PCMCIA) common memory write
 52h  (PCMCIA) force media change (DOS 5+ ???) (see #01568)
 53h  (PCMCIA) erase drive
 54h  (PCMCIA) erase media
 56h  (PCMCIA) set erase status callback
```

```
 57h  (PCMCIA) append Card Information Structure (CIS) tuple
 58h  (PCMCIA) erase CIS tuples
 60h  get device parameters (see #01560)
 61h  read logical device track (see #01562)
 62h  verify logical device track (see #01564)
 66h  (DOS 4.0+) get volume serial number (see #01565,AH=69h)
 67h  (DOS 4.0+) get access flag (see #01566)
 68h  (DOS 5.0+) sense media type (see #01569)
 6Ah  (MS-DOS 7.0) unlock logical volume (see AX=440Dh/CX=086Ah)
  no parameter block required
 6Bh  (MS-DOS 7.0) unlock physical volume (see AX=440Dh/CX=086Bh)
  no parameter block required
 6Ch  (MS-DOS 7.0) get lock flag (see AX=440Dh/CX=086Ch)
  no parameter block required
 6Dh  (MS-DOS 7.0) enumerate open files (see AX=440Dh/CX=086Dh)
 6Eh  (MS-DOS 7.0) find swap file (see AX=440Dh/CX=086Eh)
 6Fh  (MS-DOS 7.0) get drive map information (see #01570)
 70h  (PCMCIA) attribute memory read
 70h  (MS-DOS 7.0) get current lock state (see AX=440Dh/CX=0870h)
  no parameter block required
 71h  (MS-DOS 7.0) get first cluster (see AX=440Dh/CX=0871h)
 73h  (PCMCIA) get memory media information (DOS 5+ ???) (see #01572)
 76h  (PCMCIA) get erase status callback
 77h  (PCMCIA) get first Card Information Structure (CIS) tuple
 78h  (PCMCIA) get next CIS tuple
 7Fh  (PCMCIA) get ??? information (see #01573,#01574)

Format of parameter block for functions 40h, 60h:
Offset  Size  Description (Table 01560)
 00h  BYTE  special functions
    bit 0 set if function to use current BPB, clear if Device
        BIOS Parameter Block field contains new default BPB
    bit 1 set if function to use track layout fields only
      must be clear if CL=60h
    bit 2 set if all sectors in track same size (should be set)
    bits 3-7 reserved (MS-DOS, Novell DOS 7)
    bit 5: skip head settling time (WinDOS 2.11)
    bit 6: format access flag (WinDOS 2.11)
 01h  BYTE  device type (see #01561)
 02h  WORD  device attributes
    bit 0 set if nonremovable medium
```

```
    bit 1 set if door lock ("changeline") supported
    bits 2-15 reserved
 04h  WORD  number of cylinders
 06h  BYTE  media type
    for 1.2M drive
        00h 1.2M disk (default)
        01h 320K/360K disk
    F8h for DUBLDISK.SYS v2.6 expanded drives
    always 00h for other drive types
 07h 31 BYTEs device BPB (see #01663 at AH=53h), bytes after BPB offset 1Eh
      omitted; final six bytes only transferred on function 40h
      with BYTE 00h bit 0 set for MS-DOS 5.0
---function 40h only---
 26h  WORD  number of sectors per track (start of track layout field)
    (maximum 63)
 28h  N word pairs: number,size of each sector in track
---category code 48h (FAT32), function 40h---
 07h 53 BYTEs extended BPB (see #01664)
 3Ch 32 BYTEs reserved
 5Ch  WORD  number of track table entries
 5Eh 2N WORDs sector table (word pairs: number/size of each sector in track)
---category code 48h (FAT32), function 60h---
 07h 53 BYTEs extended BPB (see #01664)
 3Ch 32 BYTEs reserved

(Table 01561)
Values for device type:
 00h  320K/360K 5.25-inch floppy disk
 01h  1.2M 5.25-inch floppy disk
 02h  720K floppy disk
 03h  single-density 8-inch disk
 04h  double-density 8-inch disk
 05h  fixed disk
 06h  tape drive
 07h  (DOS 3.3+) other type of block device, normally 1.44M floppy
 08h  read/write optical disk
 09h  (DOS 5+) 2.88M 3.5-inch floppy
 FFh  (some DOS 5 betas) 2.88M 3.5-inch floppy

Format of parameter block for functions 41h, 61h:
Offset  Size  Description (Table 01562)
```

```
 00h  BYTE  special functions (reserved, must be zero)
 01h  WORD  number of disk head
 03h  WORD  number of disk cylinder
 05h  WORD  number of first sector to read/write
 07h  WORD  number of sectors
 09h  DWORD transfer address
Note: under Windows95, a volume must be locked (see AX=440Dh/CX=084Bh) in
    order to perform direct accesses such as track reads and writes
    with this IOCTL function

Format of parameter block for function 42h:
Offset  Size  Description (Table 01563)
 00h  BYTE  reserved, must be zero (DOS <3.2)
      bit 0=0: format/verify track
      1: format status call (DOS 3.2+), don't actually format
      bit 1: format multiple tracks, require additional WORD
        (hard disks only)
      bits 2-7 reserved, must be zero
    value on return (DOS 3.3+):
      00h specified tracks, sectors/track supported by BIOS
      01h function not supported by BIOS
      02h specified tracks, sectors/track not allowed for drive
      03h no disk in drive
 01h  WORD  number of disk head
 03h  WORD  number of disk cylinder
---BYTE 00h bit 1 set---
 05h  WORD  number of tracks to format

Format of parameter block for function 62h:
Offset  Size  Description (Table 01564)
 00h  BYTE  reserved, must be zero (DOS <3.2)
      bit 0=0: verify single track
      1: verify multiple tracks
      bits 1-7 reserved, must be zero
    value on return (DOS 3.3+):
      00h specified tracks, sectors/track supported by BIOS
      01h function not supported by BIOS
      02h specified tracks, sectors/track not allowed for drive
      03h no disk in drive
 01h  WORD  number of disk head
 03h  WORD  number of disk cylinder
```

```
 05h  WORD  number of tracks to verify (equivalent to 255 or fewer sectors)

Format of parameter block for functions 46h, 66h:
Offset  Size  Description (Table 01565)
 00h  WORD  (call) info level (should be 0000h)
 02h  DWORD disk serial number (binary)
 06h 11 BYTEs volume label or "NO NAME    "
 11h  8 BYTEs filesystem type "FAT12   " or "FAT16   "
      (generally CL=66h only, but MS-DOS 5.0 will write the
      given filesystem type to the disk)
Note: under MS-DOS 7.0 or a Windows95 DOS box, the volume label field can
    return as all blanks even when a volume label has been set (the Win95
    installation seems to blank the volume label field in the partition
    boot sector; once LABEL has been run, the volume label is reported
    correctly)
SeeAlso: AH=69h

Format of parameter block for functions 47h, 67h:
Offset  Size  Description (Table 01566)
 00h  BYTE  special-function field (must be zero)
 01h  BYTE  disk-access flag, nonzero if access allowed by driver

Format of parameter block for function 48h:
Offset  Size  Description (Table 01567)
 00h  BYTE  (call) locking operation
    00h lock media in drive
    01h unlock media
    02h get locking status
 01h  BYTE  (ret) drive's lock status (number of pending locks on drive)
Note: also supported by MS-DOS 7.0+

Format of parameter block for function 52h:
 00h  BYTE  (call) unused???  (Table 01568)
    (ret) 00h if flash/ATA drive but no card inserted
      unchanged otherwise
Notes:  the absense of a flash card should be tested by checking the DOS error
    code rather than the returned byte
  the parameter byte is cleared to 00h erroneously by the Award
    PCDISK.EXE v1.02c PCMCIA/ATA driver if no ATA card is inserted
    (bug corrected in PCDISK.EXE v1.02h and later)
  not supported by the SystemSoft ATADRV.EXE and the Phoenix PCMATA.SYS
```

```
    PCMCIA/ATA drivers

Format of parameter block for function 68h:
Offset  Size  Description (Table 01569)
 00h  BYTE  01h for default media type, 00h for any other media type
    (see also INT 13/AH=20h"Compaq")
 01h  BYTE  02h for 720K, 07h for 1.44M, 09h for 2.88M

Format of parameter block for function 6Fh:
Offset  Size  Description (Table 01570)
 00h  BYTE  (call) length of this buffer (in bytes)
 01h  BYTE  (ret) number of bytes in parameter block actually used
 02h  BYTE  (ret) drive flags (see #01571)
 03h  BYTE  (ret) physical drive number
    00h-7Fh floppy
    80h-FEh hard
    FFh no physical drive
 04h  DWORD (ret) bitmap of logical drives associated with physical drive
    bit 0 = drive A:, etc.
 08h  QWORD (ret) relative block address of partition start

Bitfields for Get Drive Map Information drive flags:
Bit(s)  Description (Table 01571)
 0  protected-mode driver for logical drive
 1  protected-mode driver in use for physical drive corresponding to the
    logical drive
 2  drive available only in protected mode
 3  protected-mode drive supports media ejection
 4  drive issues media insertion and removal notifications
SeeAlso: #01570

Format of parameter block for function 73h:
Offset  Size  Description (Table 01572)
 00h  BYTE  ???
    00h ATA card inserted ???
    80h ATA card not inserted ???
 01h  BYTE  length of parameter block ???
    apparently always 40h
 02h  BYTE  ???
    00h ATA card not inserted ???
    0Dh ATA card inserted ???
```

```
 03h  2 BYTEs ??? (apparently always 00h)
 05h  BYTE  drive number (0=first) ???
 06h  BYTE  total number of drives ???
 07h  BYTE  ???
    00h ATA card not inserted ???
    01h ATA card inserted ???
 08h 17 BYTEs ???
 19h  BYTE  ???
    00h ATA card not inserted ???
    01h ATA card inserted ???
 1Ah  BYTE  ??? (apparently always 01h)
 1Bh  BYTE  ???
    00h ATA card not inserted ???
    01h ATA card inserted ???
 1Ch  2 BYTEs ??? (apparently always 0015h)
 1Eh  2 BYTEs ???
 20h  2 BYTEs ??? (apparently always 0110h)
 22h 15 BYTEs ???
 31h  2 BYTEs ??? (apparently always 7000h)
 33h 11 BYTEs driver signature
    "AWARD PDISK" for Award PCDISK.EXE PCMCIA/ATA driver
    "MS-BIOS    " for HP 200LX generic ATA driver
 3Eh  2 BYTEs ???
Notes:  parameter structure possibly depends on driver
  this function is not supported by the SystemSoft ATADRV.EXE and the
    Phoenix PCMATA.SYS PCMCIA/ATA drivers

Format of parameter block for function 7Fh for SystemSoft ATADRV.EXE:
Offset  Size  Description (Table 01573)
 00h  DWORD -> unknown location within driver
Note: function supported by the SystemSoft ATADRV.EXE PCMCIA/ATA driver
    but not by the Award PCDISK.EXE PCMCIA/ATA driver
SeeAlso: #01574

Format of parameter block for function 7Fh for Phoenix PCMATA.SYS:
Offset  Size  Description (Table 01574)
 00h  8 BYTEs ???
Note: this function supported by the Phoenix PCMATA.SYS PCMCIA/ATA driver
    but not by the Award PCDISK.EXE PCMCIA/ATA driver
SeeAlso: #01573
--------D-21440DCX084A-----------------------
```

```
INT 21 - MS-DOS 7.0+ - GENERIC IOCTL - LOCK LOGICAL VOLUME
  AX = 440Dh
  CX = 084Ah / 484Ah
       (category code 08h for FAT12/16, 48h for FAT32; minor code 4Ah)
  BL = drive number (01h=A:,02h=B:,etc)
  BH = lock level (00h-04h)
  DX = drive permissions (see #01575) for Level 1 lock or second
       Level 0 lock when formatting
Return: CF set on error
       AX = error code (01h,02h,etc.) (see #01680 at AH=59h/BX=0000h)
  CF clear if successful
Notes:  the logical volume must be locked before direct disk accesses are
    permitted by Windows95/98
  the commandline LOCK issues a level 4 lock
  Windows98 only permits lock levels 0 and 4
BUG:  Windows98 will return an error (invalid function) if the specified
    drive number is zero or more than 32, but only allocated 26 bytes
    for recording locks, so BL=1Bh..20h will trash internal data
    structures
SeeAlso: AX=440Dh"DOS 3.2+",AX=440Dh/CX=084Bh,AX=440Dh/CX=086Ah
SeeAlso: AX=440Dh/CX=086Ch

Bitfields for drive permissions:
Bit(s)  Description (Table 01575)
 0  allow writes
 1  disallow new file mappings
 2  volume locked for formatting
--------D-21440DCX084B-----------------------
INT 21 - MS-DOS 7.0+ - GENERIC IOCTL - LOCK PHYSICAL VOLUME
  AX = 440Dh
  CX = 084Bh / 484Bh
       (category code 08h for FAT12/16, 48h for FAT32; minor code 4Bh)
  BH = logical drive number (00h = default, 01h = A:, etc.)
  BL = lock level (00h-03h)
  DX = drive permissions (see #01575) for Level 1 lock or second
       Level 0 lock when formatting
Return: CF set on error
       AX = error code (01h,02h,etc.) (see #01680 at AH=59h/BX=0000h)
  CF clear if successful
Desc: lock all logical volumes on the same physical volume as the indicated
    drive
```

```
Note: the physical volume must be locked before direct disk writes via
    INT 13 are permitted by Windows95
SeeAlso: AX=440Dh"DOS 3.2+",AX=440Dh/CX=084Ah,AX=440Dh/CX=086Bh
SeeAlso: AX=440Dh/CX=086Ch
--------D-21440DCX086A-----------------------
INT 21 - MS-DOS 7.0+ - GENERIC IOCTL - UNLOCK LOGICAL VOLUME
  AX = 440Dh
  CX = 086Ah / 486Ah
       (category code 08h for FAT12/16, 48h for FAT32; minor code 6Ah)
  BL = drive number (00h=default,01h=A:,etc)
Return: CF set on error
      AX = error code (01h,02h,etc.) (see #01680 at AH=59h/BX=0000h)
  CF clear if successful
Note: the logical volume must be locked before direct disk accesses via
    INT 13 are permitted by Windows95
BUG:  Windows98 will return an error (invalid function) if the specified
    drive number is zero or more than 32, but only allocated 26 bytes
    for recording locks, so BL=1Bh..20h will trash internal data
    structures
SeeAlso: AX=440Dh"DOS 3.2+",AX=440Dh/CX=084Ah,AX=440Dh/CX=086Bh
--------D-21440DCX086B-----------------------
INT 21 - MS-DOS 7.0+ - GENERIC IOCTL - UNLOCK PHYSICAL VOLUME
  AX = 440Dh
  CX = 086Bh / 486Bh
       (category code 08h for FAT12/16, 48h for FAT32; minor code 6Bh)
  BH = logical drive number (00h = default, 01h = A:, etc.)
Return: CF set on error
      AX = error code (01h,02h,etc.) (see #01680 at AH=59h/BX=0000h)
  CF clear if successful
Desc: lock all logical volumes on the same physical volume as the indicated
    drive
SeeAlso: AX=440Dh"DOS 3.2+",AX=440Dh/CX=084Bh,AX=440Dh/CX=086Ah
--------D-21440DCX086C-----------------------
INT 21 - MS-DOS 7.0+ - GENERIC IOCTL - GET LOCK FLAG STATE
  AX = 440Dh
  CX = 086Ch / 486Ch
       (category code 08h for FAT12/16, 48h for FAT32; minor code 6Ch)
  BL = drive number (00h=default,01h=A:,etc)
Return: CF set on error
      AX = error code (01h,02h,etc.) (see #01680 at AH=59h/BX=0000h)
  CF clear if successful
```

```
      AX = access flag (see #01576)
SeeAlso: AX=440Dh"DOS 3.2+",AX=440Dh/CX=084Ah,AX=440Dh/CX=084Bh
SeeAlso: AX=440Dh/CX=0870h

(Table 01576)
Values for drive access flag:
 0000h  no writes/file mappings since last call
 0001h  write operation has occurred since last call
 0002h  file mapping has occurred since last call
Note: function 6Ch resets the access flag every time it is called
--------D-21440DCX086D-----------------------
INT 21 - MS-DOS 7.0+ - GENERIC IOCTL - ENUMERATE OPEN FILES
  AX = 440Dh
  CX = 086Dh / 486Dh
       (category code 08h for FAT12/16, 48h for FAT32; minor code 6Dh)
  BL = drive number (00h=default,01h=A:,etc)
  DS:DX -> buffer for ASCIZ pathname
  SI = file index (0000h to number of open files-1)
  DI = enumeration type (0000h all files, 0001h unmovable files)
Return: CF set on error
      AX = error code (01h,02h,12h,etc.) (see #01680 at AH=59h/BX=0000h)
    0012h if file index is out of range
  CF clear if successful
      AX = file open mode (BX from AX=6C00h or AX=716Ch)
      CX = file type (see #01577)
SeeAlso: AX=440Dh"DOS 3.2+",AX=440Dh/CX=084Ah,AX=440Dh/CX=086Ch
SeeAlso: AX=440Dh/CX=086Eh,AX=6C00h,AX=716Ch

(Table 01577)
Values for file type:
 0000h  normal file
 0001h  memory-mapped file (unmovable)
 0002h  unmovable file
 0004h  swap file
--------D-21440DCX086E-----------------------
INT 21 - MS-DOS 7.0+ - GENERIC IOCTL - FIND SWAP FILE
  AX = 440Dh
  CX = 086Eh / 486Eh
       (category code 08h for FAT12/16, 48h for FAT32; minor code 6Eh)
  BL = drive number (00h=default,01h=A:,etc)
  DS:DX -> buffer for ASCIZ pathname
```

```
  SI = file index
  DI = enumeration type
Return: CF set on error
      AX = error code (01h,02h,etc.) (see #01680 at AH=59h/BX=0000h)
  CF clear if successful
      AX = pager type
    0001h no pager
    0002h paging through MS-DOS
    0003h protected-mode pager
      CX:BX = swap file size in 4K pages
SeeAlso: AX=440Dh"DOS 3.2+",AX=440Dh/CX=086Dh
--------D-21440DCX0870-----------------------
INT 21 - MS-DOS 7.0+ - GENERIC IOCTL - GET CURRENT LOCK STATE
  AX = 440Dh
  CX = 0870h / 4870h
       (category code 08h for FAT12/16, 48h for FAT32; minor code 70h)
  BL = drive number (00h=default,01h=A:,etc)
Return: CF set on error
      AX = error code (01h,02h,etc.) (see #01680 at AH=59h/BX=0000h)
  CF clear if successful
      AX = current lock level (0-3) or FFFFh if not locked
      CX = lock permissions if AX<>FFFFh (see #01575)
SeeAlso: AX=440Dh"DOS 3.2+",AX=440Dh/CX=086Ch
--------D-21440DCX0871-----------------------
INT 21 - MS-DOS 7.0+ - GENERIC IOCTL - GET FIRST CLUSTER
  AX = 440Dh
  CX = 0871h / 4871h
       (category code 08h for FAT12/16, 48h for FAT32; minor code 71h)
  BX = filename character set (see #01578)
  DS:DX -> ASCIZ pathname for file or directory
Return: CF set on error
      AX = error code (01h,02h,etc.) (see #01680 at AH=59h/BX=0000h)
  CF clear if successful
      DX:AX = first cluster number
Note: this function finds any file or directory regardless of attributes,
    except that it will not find volume labels
SeeAlso: AX=440Dh"DOS 3.2+",AX=440Dh/CX=086Ch,#01352

(Table 01578)
Values for character set:
 00h  Windows ANSI
```

```
 01h  current OEM character set
 02h  Unicode
--------y-21440DCXEDC1-----------------------
INT 21 - DR PalmDOS - GENERIC IOCTL - LOGIN SECURITY
  AX = 440Dh
  CX = EDC1h (category code EDh, minor code C1h)
  BL = boot drive number (01h=A:,02h=B:,etc)
Return: AL = 0 if already logged in
  AL <> 0 if system is still secured (not logged in)
  ES:DI -> ???
Notes:  This function is called by DR PalmDOS IBMBIO.COM after CONFIG.SYS
    processing has finished and it has launched the LOGIN.SYS utility
    (if this is not in the root of the boot drive, the undocumented
    CONFIG.SYS LOGIN directive can be used to change the path and the
    moment when the login prompt will be displayed).
  If security is active and the user has still not logged in,
    IBMBIO.COM will display an error message and halt the system.
--------D-21440E-----------------------------
INT 21 - DOS 3.2+ - IOCTL - GET LOGICAL DRIVE MAP
  AX = 440Eh
  BL = drive number (00h=default,01h=A:,etc)
Return: CF set on error
      AX = error code (01h,0Fh) (see #01680 at AH=59h/BX=0000h)
  CF clear if successful
      AL = 00h block device has only one logical drive assigned
     1..26 the last letter used to reference the drive (1=A:,etc)
Notes:  DR DOS 3.41-5.0 DRIVER.SYS does not support drive mapping and thus
    always returns AL=00h; Windows NT 4.0 also always returns AL=00h
  in addition to the normal operation, if Stacker is installed, this
    call also sets the volume number at offset 58h in the Stacker
    device driver (DR DOS 3.41-5.0 only; use AX=4408h otherwise)
    (see AX=4404h"Stacker",INT 25/AX=CDCDh)
SeeAlso: AX=4408h,AX=440Fh,INT 2F/AX=122Bh
--------D-21440F-----------------------------
INT 21 - DOS 3.2+ - IOCTL - SET LOGICAL DRIVE MAP
  AX = 440Fh
  BL = physical drive number (00h=default,01h=A:,etc))
Return: CF set on error
      AX = error code (01h,0Fh) (see #01680 at AH=59h/BX=0000h)
  CF clear if successful
      drive now responds to next logical drive number
```

```
Notes:  maps logical drives to physical drives, similar to DOS's treatment of
    a single physical floppy drive as both A: and B:
  DR DOS 3.41-5.0 DRIVER.SYS does not support drive mapping and thus
    always returns an error on this function
SeeAlso: AX=440Eh,INT 2F/AX=122Bh
--------D-214410----------------------------
INT 21 - DOS 5+ - IOCTL - QUERY GENERIC IOCTL CAPABILITY (HANDLE)
  AX = 4410h
  BX = handle for device
  CH = category code (see #01545)
  CL = function code (see #01546)
Return: CF clear if successful
      AX = 0000h  specified IOCTL function is supported
  CF set on error
      AL = 01h  IOCTL capability not available
Note: a program which wishes to use Generic IOCTL calls beyond those in the
    standard DOS 3.2 set may use this call first to see whether a
    particular call is supported
SeeAlso: AX=440Ch,AX=440Dh"DOS 3.2+",AX=4411h
--------d-214410BXFFFF----------------------
INT 21 U - NewSpace - ENABLE DRIVER
  AX = 4410h
  BX = FFFFh
Program: NewSpace is a TSR by Isogon Corporation which automatically compresses
    all files as they are written and decompresses them as they are read
Note: compressed files are not accessible unless the driver is enabled
SeeAlso: AX=4411h/BX=FFFFh
--------D-214411----------------------------
INT 21 - DOS 5+ - IOCTL - QUERY GENERIC IOCTL CAPABILITY (DRIVE)
  AX = 4411h
  BL = drive number
  CH = category code (see #01558)
  CL = function code (see #01559)
Return: CF clear if successful
      AX = 0000h  specified IOCTL function is supported
  CF set on error
      AL = 01h  IOCTL capability not available
Note: a program which wishes to use Generic IOCTL calls beyond those in the
    standard DOS 3.2 set may use this call first to see whether a
    particular call is supported
SeeAlso: AX=440Ch,AX=440Dh"DOS 3.2+",AX=4410h
```

```
--------d-214411BXFFFF-----------------------
INT 21 U - NewSpace - DISABLE DRIVER
  AX = 4411h
  BX = FFFFh
Program: NewSpace is a TSR by Isogon Corporation which automatically compresses
    all files as they are written and decompresses them as they are read
Note: compressed files are not accessible unless the driver is enabled
SeeAlso: AX=4410h/BX=FFFFh
--------O-214412-----------------------------
INT 21 O - DR DOS 5.0-6.0 - DETERMINE DOS TYPE
  AX = 4412h
  CF set
Return: CF set if not DR DOS
      AX = error code (see #01680 at AH=59h/BX=0000h)
  CF clear if DR DOS
      DX = AX = version code (see #01580)
Note: this obsolete call, which is no longer supported in Novell DOS 7, is
    identical to AX=4452h
SeeAlso: AX=4452h
--------d-214412BXFFFF-----------------------
INT 21 U - NewSpace - INSTALLATION CHECK???
  AX = 4412h
  BX = FFFFh
Return: AX = PSP segment of NewRes (resident driver for NewSpace)
  BX:DX -> ???
  CX = ???
SeeAlso: AX=4411h/BX=FFFFh
--------d-214413BXFFFF-----------------------
INT 21 U - NewSpace - GET ???
  AX = 4413h
  BX = FFFFh
Return: AX = code segment of NewRes (resident driver for NewSpace)
  BX = offset of ???
SeeAlso: AX=4412h/BX=FFFFh
--------O-214414-----------------------------
INT 21 OU - DR DOS 5.0-6.0 - SET GLOBAL PASSWORD
  AX = 4414h
  DS:DX -> password string (blank-padded to 8 characters)
Desc: Specify the master password for accessing files.
Note: this obsolete call, which is no longer supported in Novell DOS 7, is
    identical to AX=4454h
```

```
SeeAlso: AX=4454h
--------d-214414BXFFFF----------------------
INT 21 U - NewSpace - DEBUGGING DUMP
  AX = 4414h
  BX = FFFFh
Return: debugging dump written to X:\NEWSPACE.SMP
SeeAlso: AX=4413h/BX=FFFFh,AX=44FFh/BX=FFFFh
--------O-214416----------------------------
INT 21 OU - DR DOS 5.0-6.0 - HISTORY BUFFER, SHARE, AND HILOAD CONTROL
  AX = 4416h to 4418h
Note: these obsolete subfunctions (which are no longer supported in Novell
    DOS 7) are identical to AX=4456h through 4458h
SeeAlso: AX=4456h,AX=4457h,AX=4458h
--------O-214451----------------------------
INT 21 - Concurrent DOS v3.2+ - INSTALLATION CHECK
  AX = 4451h
Return: CF set if not Concurrent DOS
      AX = error code (see #01680 at AH=59h/BX=0000h)
  CF clear if successful
      AH = single-user/multiuser nature (see #01581)
    10h single-user
        AL = operating system version ID (see #01580)
    14h multiuser
        AL = operating system version ID (see #01579)
Notes:  as of Concurrent DOS/XM 5.0 (possibly earlier), the version is stored
    in the environment variable VER and the OS type in variable OS
  use this function if you are looking for multiuser capabilities,
    AX=4452h for single-user
  this function should never return the single-user values; however, an
    application should be prepared to accept single-user values, i.e. it
    should check the returned AH and AL separately
  CCI Multiuser DOS 7.22 returns AX=DX=1466h.
  DR DOS 3.31+ error returns with AX=0001h
SeeAlso: AX=4452h,AX=4459h

(Table 01579)
Values for Digital Research operating system version ID:
 32h  Concurrent PC DOS 3.2
 41h  Concurrent DOS 4.1
 50h  Concurrent DOS/XM 5.0 or Concurrent DOS/386 1.1
 60h  Concurrent DOS/XM 6.0 or Concurrent DOS/386 2.0
```

```
 62h  Concurrent DOS/XM 6.2 or Concurrent DOS/386 3.0
 66h  DR Multiuser DOS 5.1, CCT Multiuser DOS 7.x
 67h  Concurrent DOS 5.1
SeeAlso: #01581,#04023
--------O-214452-----------------------------
INT 21 - DR DOS 3.41+ - DETERMINE DOS TYPE/GET DR DOS VERSION
  AX = 4452h ("DR")
  CF set
Return: CF set if not DR DOS
      AX = error code (see #01680 at AH=59h/BX=0000h)
  CF clear if DR DOS
      AX = version code
      AH = single-user/multiuser nature (see #01581)
    10h single-user
        AL = operating system version ID (see #01580)
    14h multiuser
        AL = operating system version ID (see #01579)
      DX modified (refer to note below)
Notes:  the DR DOS version is stored in the environment variable VER
  use this function if looking for single-user capabilities, AX=4451h
    if looking for multiuser; this call should never return multiuser
    values
  in DR DOS 3.41-6.0, DX=AX on return; for Novell DOS 7, DH=AH but DL=00h
    (reportedly, DH=00h when booting NWDOS7 from installation disks)
  Novell DOS 7 returns error code 0001h if SETVER x.255 is in effect for
    the calling program, or SETVER /G x.255 is in effect
  although based on DR DOS, CCI Multiuser DOS 7.xx,
    IMS Multiuser DOS 7.x, and IMS REAL/32 7.50+ do not support this call
  With OpenDOS 7.02 BETA 2 and DR-OpenDOS 7.02+, the install check
    in most tools has been changed to run with both, AH=10h and AH=14h
    to support possible future "client side" multiuser releases of
    DR-DOS which may return 14h instead of 10h. Tools should also strip
    off the CP/Net bit before checking the kernel version.
    Often used version checks like >= 1070h are invalid, therefore,
    hence the AH and AL must be checked seperately.
  In DR DOS 3.31-6.0, and DR PalmDOS, DX=AX on return.
  For Novell DOS 7 - DR DOS 7.03 the DX value returned by this function
    is the same as the DX value returned by INT 21/AX=3306h, and
    represents the DOS revision (bits 7-0, currently always DL=00h) and
    the version flags (bits 15-8, with bit 12 = DOS in HMA, bit 11 = DOS
    in ROM) of the "patch_version" field in the PCM_HEADER structure in
```

the IBMDOS.COM file. The version flags, however, are updated at runtime to reflect the actual system status, resulting, for example, in DH=10h for DOS in HMA, and DH=0 when booting Novell DOS 7 from installation disks.

The operating system version ID represents the BDOS (Basic Disk Operating System) kernel version (of the DRBDOS.SYS aka IBMDOS.COM file), and the two nibbles can actually be read as CP/M version and sub-version, that is DR-DOS 7.03 (version code 1073h) is actually CP/M-86 version 7.3.

Due to lost original patch sources from the backups, Caldera OpenDOS 7.01 merely represented Novell DOS 7 Update 10 with minor changes.

For Caldera DR-OpenDOS 7.02, all the Novell DOS 7 patches up to including Update 15.2 have been re-incorporated into the system from other sources, while the missing patches for IBMBIO.COM were independently re-implemented by Matthias Paul in 07/1997-10/1997.

Novell DOS 7 (since 1993-11-08), OpenDOS 7.01 and OpenDOS 7.02 BETA IBMDOS.COM return error code 0001h if SETVER x.255 is in effect for the calling program, or SETVER /G x.255 is in effect.

Since Caldera DR-OpenDOS 7.02 the SHARE 2.05+ (1998-01-05) driver uses INT 21/AX=65A2h as an *additional* BDOS version check: If the "FUCASE char" function is functional on a DR-DOS BDOS kernel 72h+, SHARE assumes it is running on an OpenDOS 7.02 BETA 1 (73h) kernel (or later) even if the BDOS version returned by INT 21/AX=4452h would indicate an earlier issue of the kernel. Some 3rd party drivers (like HPFS_DOS.EXE) were hardwired to detect Novell DOS 7 only, and so the kernel version had to be patched back on such systems to allow such drivers to work properly. The BDOS version and DOS revision are stored in PCM_HEADER of the IBMDOS.COM file, see table XXXX below.

However, this is no longer necessary with the introduction of the DR-DOS 7.02+ IBMDOS.COM (since 1998-01-10) and SETVER 1.01+ (since 1998-01-12) because they allow to fake BDOS versions as well as faking DOS versions. In /X mode, a set sub-version of y = 100..127 will be used as BDOS version instead (64h..7Fh), while the DOS revision will be used to report the DOS sub-version instead. For example, given a DOS revision of 0, SETVER 6.114 would be the proper value to fake Novell DOS 7 (114=72h) on a DR-DOS 7.02+ system, reporting a DOS version of "IBM" 6.00. Sub-versions 128..255 will completely disable this BDOS version check, and report a DOS sub-version of 0..127. This is to work around problems with possibly hostile programs, that try to detect DR-DOS to not run on this OS.

```
  Additional version check for 1072h kernels: At API level, there
    is no known way to distinguish Novell DOS 7 from OpenDOS 7.01,
    however, an IBMBIO.COM/IBMDOS.COM file date since 1997 and the
    existance of an environment variable %OS%=OPENDOS usually indicates
    an OpenDOS system (but not the other way around; some OpenDOS
    BETAs still used %OS%=NWDOS!).
  Additional version checks for 1073h kernels: A functional test of
    INT 21/AX=65A2h (see above for SHARE) can be used to distinguish
    OpenDOS 7.02 BETA 1 from the later (OpenDOS 7.02 BETA 2+ and)
    DR-OpenDOS 7.02 and DR-DOS 7.02 releases, which are both the
    same, except for the name change and minor bug fixes. Testing
    for the INT 21/AX=6601h/BX=0000h bug can be used to differentiate the
    original release of DR-DOS 7.02 from later 7.02 updates and
    DR-DOS 7.03.
  The DR DOS version is stored in the environment variable VER.
SeeAlso: AX=4412h,AX=4451h,AX=4459h

(Table 01580)
Values for Digital Research operating system version ID:
 41h  DOS Plus 1.2
 60h  DOS Plus 2.0 or 2.1
 63h  DR DOS 3.41
 64h  DR DOS 3.42
 65h  DR DOS 5.0
 67h  DR DOS 6.0
 70h  PalmDOS
 71h  DR DOS 6.0 March 1993 "business update"
 72h  Novell DOS 7.0
 73h  OpenDOS 7.02
SeeAlso: #01579,#01581,#04088,#04023

(Table 04088)
Values for Digital Research operating system ID codes (full AX return value):
 ??41h  - DOS Plus 1.2
 1060h  - DOS Plus 2.0 or 2.1 (1988-03-09???)
    (this was basically a Concurrent CP/M-86 with built-in DOS emulation
     developed between 1983-1986)
 1060h  - DR DOS 3.31 (OEM) (1988-04-27???, 06/1988)
  - DR DOS 3.32 (OEM) (1988-08-17)
  - DR DOS 3.33 (OEM) (1988-09-01)
  - DR DOS 3.34 (OEM)
```

```
 - DR DOS 3.35 (OEM) (1988-10-21)
   for example: at least an issue for "Prism System 232" (10/1988)
   and a French issue existed.
106?h  - DR DOS 3.40 BETA 1 (1988-11-11)
 - DR DOS 3.40
   (This is the first issue for sure known to be ROMmable, although
   DR DOS should have been ROMmable right from the start (1987).)
1063h  - DR DOS 3.41 various OEM and retail releases (06/1989-07/1989)
 - DR DOS 3.41T
1064h  - DR DOS 3.42
   (It is not clear, if this issue was actually released, since Lineo
   recently (fall 1999) released some historical kernels, and one of
   them is sailing under the name DR DOS 5.0 "Leopard" BETA 1 and has
   copyright strings from 1990, while it still identifies itself as
   being DR DOS 3.41... Unfortunately the uploaded archive is broken,
   hence it is not possible to find out easily. Maybe DR DOS 3.42 was
   nothing more than this early "Leopard" BETA???)
1065h  - DR DOS ??? (1990-02-01)
 - DR DOS 5.0 "Leopard" BETA 2 (1990-03-16),
 - DR DOS 5.0 "Leopard" BETA 2B
   (This was the first issue to use the new IBMBIO.COM/IBMDOS.COM
   names instead of DRBIOS.SYS/DRBDOS.SYS.)
 - DR DOS 5.0 "Leopard" BETA 3
   (This issue still uses separate boot sectors for floppies and
   harddisks. The boot sector OEM label is still "DIGITAL".)
 - DR DOS 5.0 release (06/1990, 08/1990)
   (This issue has a combined boot sector for both, floppies and
   harddisks. The boot sector OEM label has changed to "IBM  3.3".
   DR DOS 5.0 was the first DOS to introduce high-load capabilities.)
1066h  - DR DOS ??? (1990-08-09)
 - DR DOS 6.0 "Buxton" ALPHA (02/1991-03/1991)
1067h  - DR DOS ??? (1991-03-15)
106?h  - DR DOS 6.0 "Buxton" BETA 2 (??/1991)
106?h  - DR DOS 6.0 "Buxton" BETA 3 (05/1991, 1991-06-27, 1991-06-28)
1067h  - DR DOS 6.0 release (05/1991, 08/1991)
 - DR DOS 6.0 BDOS patch "PAT304" English (1991-11-29, XDIR /C: 9D8Ch)
   fix for "\\" in cmdline by Quick Basic 4.5/MS PDS 7.1 on Lantastic 4.0
 - DR DOS 6.0 BDOS patch "PAT303" English (1991-12-03, XDIR /C: 66B0h)
   This patch makes INT 21/AH=4Eh (Search First) compatible with MS-DOS
   when the Volume Label attribute is set. Previously under DR DOS the
   volume label was only searched for if bit 3 was the only bit set in
```

```
   the attribute whereas MS-DOS searches for the volume label if bit 3
   is set, irrespective of any other bits in the attribute.
- DR DOS 6.0 BDOS patch "PAT306" English (1991-12-11, XDIR /C: 8DE5h)
   This patch fixes a problem with OPTASM which would give error 8
   if there were not enough handles available. This patch forces the
   system to check for available handles before it opens a file.
- DR DOS 6.0 BDOS patch "PAT312" English (1992-01-07, XDIR /C: A0C6h)
   This patch fixes a problem with INT 21/AH=26h (create PSP) the
   segment from which the PSP was copied was incorrect so the the PSP
   was not filled correctly and did not contain the command tail.
- DR DOS 6.0 BDOS patch "PAT314" English (1992-01-10, XDIR /C: C964h)
   This patch modifies INT 21/AX=33xxh, the Ctrl Break handler to
   support undocumented MS-DOS function INT 21/AX=3302h.
- DR DOS 6.0 BIOS patch "PAT315" English (1992-01-10, XDIR /C: DBAAh)
   This patch fixes a problem where, when booting from a Bernoulli
   drive, any DEVICE= statement in CONFIG.SYS failed if it was placed
   after the device driver RCD.SYS v7.x and DOSOAD.SYS v1.x
- DR DOS 6.0 BDOS patch "PAT317" English (1992-01-27, XDIR /C: B701h)
   This patch fixes a problem where attempting to close a changed file
   on a write protected disk seems to work after the first retry.
   This was caused because when the retry is attempted the file has been
   marked as not changed, so the attempt to write to disk is not made.
- DR DOS 6.0 BDOS patch "PAT321" English (1992-02-19, XDIR /C: 947Bh)
   This patch fixes a problem where the error codes returned from
   INT 25h and INT 26h for "drive not ready" and "write protect"
   errors were incorrect.
- DR DOS 6.0 BDOS patch "PAT323" English (1992-02-20, XDIR /C: E1B0h)
   This patch fixes a problem where, if the first command entered after
   booting the machine is a single character, any use of the command
   line recall keys will only recall the first command.
- DR DOS 6.0 ??? German (1992-02-21)
- DR DOS 6.0 update (02/1992)
- DR DOS 6.0 BDOS patch "PAT326" English (1992-03-05, XDIR /C: 8EF2h)
   This is an optional patch to prevent DRDOS from padding the
   environment of EXEPACKed applications or utilities.
- DR DOS 6.0 ??? German (1992-03-27)
- DR DOS 6.0 BDOS patch "PAT334" English (1992-03-27, XDIR /C: 2AFEh)
   This patch fixes a problem where selecting (R)etry after hitting ^P
   while the printer is offline caused the system to hang.
- DR DOS 6.0 "Windows 3.1 update, April 1992" "dr6win" (03/1992,
   1992-04-07). This public update only includes patches addressing
```

```
   full Windows 3.1 compatiblity. There should have been a full
   "business update" for registered users, shipping a little bit later.
 - DR DOS 6.0 ??? English (1992-04-07)
 - DR DOS 6.0 BDOS patch "pat354" English (1992-07-28, XDIR /C: 3398h)
   This patch for Beame and Whiteside Networks: On opening a duplicate
   file handle which describes a device, a device open call is made to
   inform the device driver that a new file handle has been opened.
   This patch is current and up to date as of 1992-11-10.
 - DR DOS 6.0 update (07/1992)
 - DR DOS 6.0 BDOS patch "PAT301" English (1992-10-28, XDIR /C: 959Bh)
   This patch fixes a problem with apps that opens a file, with Share
   loaded, but then cannot delete the file until the file is closed.
   This causes "Money Manager" to fail.
 - DR DOS 6.0 update (11/1992)
 Note: There is no known method to distinguish these different kernels
        at runtime, except for - maybe - checksumming the IBMBIO.COM/
        IBMDOS.COM files.
       The listed patches only address a single problem, while the
        listed updates are full rebuilds, also including all previous
        fixes.
1070h  - multitasking "Vladivar" kernel architecture and introduction of
   DOS-like structures (1991-07-26, 08/1991)
 - DR DOS ??? (1991-09-26)
 - DR PalmDOS "Merlin" BETA 1-2
 - DR PalmDOS "Merlin" BETA 3 (1992-06-23)
 - DR PalmDOS "Merlin" Evaluation Release (1992-08-25)
 - DR PalmDOS Release Candidates 1-2 (1992-08-26)
 - DR PalmDOS/NetWare PalmDOS 1.0 (06/1992-11/1992)
   (DR PalmDOS was the first DR DOS re-incarnation that supported
   a genuine CDS, but still pretended to be DOS 3.31. Much of the
   former CP/M stuff was stripped out to reduce the code size to meet
   early Palmtop PC requirements. It introduced the patented dynamic
   idle detection method (by Roger Gross), a special issue of the
   TASKMAX program switcher (MINIMAX) to support Personal Information
   Modules (PIM) plug-in executables, supported Flash/ROM disk,
   XIP "Execute in Place" applications, and came with PCMCIA Card
   Services and Service Stack (SS) partially written by Ian Cullimore
   (ex-Poquet Computer Corporation founder).
 - DR DOS "Panther" BETA 1 (1992-10-16)
   (This issue already supported DPMS, had the "Vladivar" pre-emptive
   multitasker, DPMS-SuperStor disk compression, Multiuser-security
```

```
    (World/Group/Owner), and much improved multi-windowing ViewMAX/3 GUI
    which looked alot more like Windows than GEM. It pretended to be
    PC DOS 5. However, this issue was never released and later partially
    merged into the Novell DOS 7 project (without Multiuser-security and
    ViewMAX/3). In 1999, Caldera Thin Clients released ViewMAX/3 under
    GPL, see http://www.gemworld.com.)
 1070h??? - DR DOS "StarTrek" (STDOS) (07/1992-05/1993)
    (A never released, though reportedly fully functional result of
    a Novell and Apple team-up utilizing the DR DOS "Vladivar" multi-
    tasker to run Apple's Intel-port of MacOS 7.1 on an issue of DR DOS
    "Panther", even emulating HFS and Mac resources on DOS FAT file-
    systems.)
 1071h  - DR DOS ??? (1992-11-26)
  - DR DOS 6.0 "business update" "DRDOS493" English for
    Windows for Workgroups 3.1 (1993-03-19)
  - DR DOS 6.0 patch "PATDR6" English (1993-03-19) for NetWare
  - DR DOS 6.0 patch "PAT394" English (1993-09-17)
  - Apparently also some issues of DR DOS "Panther"???
 1072h  - Novell DOS 7/PNW "Smirnoff" BETA 1 (??/1993, probably 1993-04-26)
  - Novell DOS 7 "Panther" BETA 2 (08/1993)
    (This "Panther" issue is much different from the early 10/1992
    "Panther" BETA 1.)
  - Novell DOS 7 "Panther/Smirnoff" BETA 2 (08/1993)
  - Novell DOS 7 "Panther/Smirnoff" BETA 3 (09/1993)
    (This issue does not have workarounds for Windows 3.1 "AARD" code)
  - Novell DOS 7 "Panther/Smirnoff" BETA 4 (10/1993)
  - Novell DOS 7 code freeze 1993-12-03???)
  - Novell DOS 7 release (12/1993-04/1994)
    (also known as NWDOS or ND7, sometimes erroneously called NDOS)
  - Novell DOS 7 German release (1994-02-22)
    (This issue is known to have workarounds for Windows 3.1 "AARD" code)
  - Novell DOS 7 updates 1-3 (???/1994)
  - Novell DOS 7 update 4 (1994-05-31)
  - Novell DOS 7 updates 5-9 (???/1994)
  - Novell DOS 7 update 10 (12/1994)
  - Novell DOS 7 update 11 (01/1995)
  - Novell DOS 7 update 12 (04/1995)
  - Novell DOS 7 update 13 (05/1995)
  - Novell DOS 7 update 14 (08-09/1995)
  - Novell DOS 7 update 15 (12/1995)
  - Novell DOS 7 update 15.2 (01/1996)
```

```
    Note: The Novell DOS 7 updates 4-15.2 are full rebuilds, not patched
     binaries.
  - Caldera OpenDOS 7.01 BETA (02/1997)
    (basically representing Novell DOS 7 update 10 with minor
    changes here and there)
  - Caldera OpenDOS 7.01 (02-03/1997)
  - Caldera OpenDOS 7.01 M.R.S. (05/1997)
    (release of the kernel source code)
  - Matthias Paul's enhanced OpenDOS 7.02 ALPHA 1-4 kernels (07-11/1997)
 1073h  - Caldera OpenDOS 7.02 BETA 1 (09/1997)
    (incorporating most the Novell DOS 7 update 15.2 changes,
    except for IBMBIO.COM changes, plus a few other enhancements)
  - Caldera OpenDOS 7.02 BETA 2 (11/1997)
    (now also incorporating Matthias Paul's ALPHA 4 enhancements,
    the Novell DOS update 15.2 IBMBIO.COM patches have been
    re-implemented)
  - Caldera OpenDOS 7.02 BETA 2A (12/1997)
    (now incorporating all Novell DOS 7 update 15.2 changes)
  - Caldera DR-OpenDOS 7.02 (1997-12-2x)
    (now for the first time the product name is written with a hyphen!!!)
  - Caldera DR-DOS 7.02 internal build (02/1998)
  - Caldera DR-DOS 7.02 release (03/1998)
  - Caldera DR-DOS 7.02A release (06/1998)
  - DR-DOS 7.02 Update 1 (08/1998)
  - DR-DOS 7.02 Update 2 (09/1998)
  - Caldera (Thin Clients) DR-DOS 7.03 BETA (1998-12-03),
    sometimes referred to as "DR-DOS 7.03 BETA 3"
  - Caldera (Thin Clients) DR-DOS 7.03 release (1998-12-24, 1999-01-06)
  - Lineo DR-DOS 7.03 release (1999-06-07, 1999-09-09)
    (this issue has no functional changes compared to the 1999-01-06
    issue of DR-DOS 7.03)
SeeAlso: #01580,#01579,#01581,#04023

Bitfields for CP/M type indicator:
Bit(s)  Description (Table 01581)
 7-4  CPU type
  0000 = 8080
  0001 = 8086
 3-0  OS type
  0000 = CP/M
  0001 = MP/M
```

```
  0002 = CP/Net
  0004 = multiuser
Notes:  Usually 10h (single-user) or 14h (multi-user).
  While earlier releases of the system utilities performed a test for
    a version code of (>)= (10)xxh, this was changed with
    DR-OpenDOS 7.02+ (now taking a possible multi-user version into
    account): Most utilities now test for AH being either 10h or 14h,
    and a BDOS version >=xxh to allow them to properly run on possible
    future multi-user releases of DR-DOS. Before doing this test, some
    of them strip off the CP/Net bit.
SeeAlso: #01580
--------O-214454-----------------------------
INT 21 U - DR DOS 3.41+ - SET GLOBAL PASSWORD
  AX = 4454h
  DS:DX -> password string (blank-padded to 8 characters)
Desc: Specify the master password for accessing files.
SeeAlso: AX=4303h,AX=4414h
--------O-214456-----------------------------
INT 21 U - DR DOS 5.0+ - HISTORY BUFFER CONTROL
  AX = 4456h
  DL = control flags (see #01582)
Return: AL = previous value of state flags (see #01583)
Note: DR DOS 6.0 only checks bit 0 and ignores the rest of DL
SeeAlso: #01584

Bitfields for control flags:
Bit(s)  Description (Table 01582)
 0  whose buffer: 0=application, 1=COMMAND.COM
---Novell DOS 7---
 1  toggle HISTORY usage
 2  toggle INSERT state
Note: only one bit at a time may be used

Bitfields for state flags:
Bit(s)  Description (Table 01583)
 0  HISTORY buffer enabled
 1  INSERT enabled
 2-5  unused
 7  whose buffer: 0=application, 1=COMMAND.COM
--------O-214457-----------------------------
INT 21 U - DR DOS 5.0-6.0 - SHARE/HILOAD CONTROL
```

```
  AX = 4457h
  DH = subfunction
       00h enable/disable SHARE
     DL = 00h disable
        = 01h enable
        else Return: AX = ???
       01h get HILOAD status
     Return: AX = status
           0000h off
           0001h on
       02h set HILOAD status
     DL = new state (00h off, 01h on)
     Return: AX = ???
       other
     Return: AX = ???
Note: This was seen called by COMMAND.COM of DR DOS 6.0; it does not seem
    to be supported by Novell DOS 7
SeeAlso: AX=4457h/DX=FFFFh
--------O-214457DXFFFF-----------------------
INT 21 OU - DR DOS 6.0 - GET SHARE STATUS
  AX = 4457h
  DX = FFFFh
Return: AX = SHARE status
Note: not supported by Novell DOS 7
SeeAlso: INT 2F/AX=1000h
--------O-214458-----------------------------
INT 21 U - DR DOS 5.0+ internal - GET POINTER TO INTERNAL VARIABLE TABLE
  AX = 4458h
Return: ES:BX -> internal variable table (see #01584,#01585)
  AX = ??? (0B50h for DR DOS 5.0, 0A56h for DR DOS 6.0, 0FE4h for
       Novell DOS 7)
SeeAlso: AX=4452h

Format of DR DOS 5.0-6.0 internal variable table:
Offset  Size  Description (Table 01584)
 00h  WORD  ???
 02h  WORD  segment of ???
 04h  WORD  offset within DOS data segment of history control structure
       for COMMAND.COM history buffer (see #01586)
 06h  WORD  offset within DOS data segment of history control structure
       for application history buffer (see #01586)
```

```
 08h  BYTE  initial history state flags (see #01583)
 09h  2 BYTEs ???
 0Bh  WORD  KB of extended memory at startup
 0Dh  BYTE  number of far jump entry points
 0Eh  WORD  segment containing far jumps to DR DOS entry points (see #01587)
 10h  WORD  (only if kernel loaded in HMA) offset in HMA of first free HMA
      memory block (see #01588) or 0000h if none; segment is FFFFh
 12h  WORD  pointer to segment of environment variables set in CONFIG,
      or 0000h if already used
---DR DOS 6.0---
 14h  WORD  (only if kernel loaded in HMA) offset in HMA of first used HMA
    memory block (see #01588) or 0000h if none; segment is FFFFh
 16h  8 BYTEs ???
 1Eh  WORD  offset in DOS data segment of full COUNTRY.SYS filename
 20h  8 BYTEs ???
 28h  WORD  offset in DOS data segment of SHARE hook table
 2Ah  2 BYTEs ???
 2Ch  WORD  offset in DOS data segment of far pointer to INT 2F/AX=1000h
      handler
Notes:  the segment used for the DR DOS 6.0 CONFIG environment variables
    (excluding COMSPEC, VER and OS) is only useful for programs/drivers
    called from CONFIG.SYS. The word is set to zero later when the area
    is copied to the COMMAND.COM environment space.  This allows
    CONFIG.SYS to pass information to AUTOEXEC.BAT.
  the Novell DOS 7 KEYB uses offsets 10h,14h, and 2Ch in the same was
    as for DR DOS 6.0, so it is likely that the entire table is the same

Format of Novell DOS 7 internal variable table:
Offset  Size  Description (Table 01585)
 00h  ???
 1Eh  WORD  offset of COUNTRY.SYS filename
 42h 16 DWORDs  pointers to ??? entry points
  ???

Format of history control structure:
Offset  Size  Description (Table 01586)
 00h  WORD  segment of buffer
 02h  WORD  size of buffer in bytes
 04h  WORD  ???

Format of kernel entry jump table for DR DOS 5.0-6.0:
```

```
Offset  Size  Description (Table 01587)
 00h  5 BYTEs far jump to kernel entry point for CP/M CALL 5
 05h  5 BYTEs far jump to kernel entry point for INT 20
 0Ah  5 BYTEs far jump to kernel entry point for INT 21
 0Fh  5 BYTEs far jump to kernel entry point for INT 22 (RETF)
 14h  5 BYTEs far jump to kernel entry point for INT 23 (RETF)
 19h  5 BYTEs far jump to kernel entry point for INT 24
 1Eh  5 BYTEs far jump to kernel entry point for INT 25
 23h  5 BYTEs far jump to kernel entry point for INT 26
 28h  5 BYTEs far jump to kernel entry point for INT 27
 2Dh  5 BYTEs far jump to kernel entry point for INT 28
 32h  5 BYTEs far jump to kernel entry point for INT 2A (IRET)
 37h  5 BYTEs far jump to kernel entry point for INT 2B (IRET)
 3Ch  5 BYTEs far jump to kernel entry point for INT 2C (IRET)
 41h  5 BYTEs far jump to kernel entry point for INT 2D (IRET)
 46h  5 BYTEs far jump to kernel entry point for INT 2E (IRET)
 4Bh  5 BYTEs far jump to kernel entry point for INT 2F
Notes:  all of these entry points are indirected through this jump table
    to allow the kernel to be relocated into high memory while leaving
    the actual entry addresses in low memory for maximum compatibility
  some of these entry points (22h,23h,24h,2Eh,2Fh) are replaced as soon
    as COMMAND.COM is loaded, and return immediately to the caller, some
    returning an error code (the original handler for INT 2F returns
    AL=03h [fail]).

Format of HMA Memory Block (DR DOS 6.0 kernel loaded in HMA):
Offset  Size  Description (Table 01588)
 00h  WORD  offset of next HMA Memory Block (0000h if last block)
 02h  WORD  size of this block in bytes (at least 10h)
 04h  BYTE  type of HMA Memory Block (interpreted by MEM)
    00h system
    01h KEYB
    02h NLSFUNC
    03h SHARE
    04h TaskMAX
    05h COMMAND
 05h  var TSR (or system) code and data. DR DOS TSR's, such as KEYB,
      hooks interrupts using segment FFFEh instead FFFFh.
--------O-214459-----------------------------
INT 21 - DR Multiuser DOS 5.0 - API
  AX = 4459h
```

```
  CL = function (see #04019 at INT E0"CP/M-86")
  DS,DX = parameters
Notes:  DR DOS 5.0 and Novell DOS 7 return CF set and AX=0001h
  this API is also available on INT E0
SeeAlso: AX=4452h,INT E0"CP/M-86"
--------O-21445A----------------------------
INT 21 U - Concurrent DOS etc. - USER GROUP SUPPORT
  AX = 445Ah
  CX = operation type
       00h get default file access rights
       01h set default file access rights
       02h get user/group
       03h get security version
       04h set security key (needs key)
       05h set user/group (needs key)
  DX = data for operation if set
  BX = key (if required)
Return: AX = result
Note: This function was introduced on 1990/06/04 for CDOS.  It has been
    supported by DR PalmDOS and DR DOS "Panther" (BDOS 1070h),
    but it is not supported by Novell DOS 7 - DR-DOS 7.03.
--------v-2144A0----------------------------
INT 21 - VIRUS - "Horns" - INSTALLATION CHECK
  AX = 44A0h
Return: AH = FFh if installed
SeeAlso: AX=4243h"VIRUS",AX=4B04h"VIRUS"
--------N-2144E0----------------------------
INT 21 U - Sun PC-NFS - API???
  AX = 44E0h
  DS:DX -> ???
  SS:BP -> stack frame (see #01589)
Return: ???
Note: this function is also supported by Beame&Whiteside's BWPCNFS shim; the
    description presented here was derived from that shim

Format of PC-NFS stack frame:
Offset  Size  Description (Table 01589)
 00h  WORD  -> previous stack frame
 02h  DWORD return address
--------d-2144FFBXFFFF----------------------
INT 21 U - NewSpace - ???
```

```
  AX = 44FFh
  BX = FFFFh
  DX = ???
Program: NewSpace is a TSR by Isogon Corporation which automatically compresses
    all files as they are written and decompresses them as they are read
SeeAlso: AX=4414h/BX=FFFFh
--------D-2145------------------------------
INT 21 - DOS 2+ - "DUP" - DUPLICATE FILE HANDLE
  AH = 45h
  BX = file handle
Return: CF clear if successful
      AX = new handle
  CF set on error
      AX = error code (04h,06h) (see #01680 at AH=59h/BX=0000h)
Notes:  moving file pointer for either handle will also move it for the other,
    because both will refer to the same system file table
  for DOS versions prior to 3.3, file writes may be forced to disk by
    duplicating the file handle and closing the duplicate
SeeAlso: AH=3Dh,AH=46h
--------D-2146------------------------------
INT 21 - DOS 2+ - "DUP2", "FORCEDUP" - FORCE DUPLICATE FILE HANDLE
  AH = 46h
  BX = file handle
  CX = file handle to become duplicate of first handle
Return: CF clear if successful
  CF set on error
      AX = error code (04h,06h) (see #01680 at AH=59h/BX=0000h)
Notes:  closes file with handle CX if it is still open
  DOS 3.30 hangs if BX=CX on entry
  moving file pointer for either handle will also move it for the other,
    because both will refer to the same system file table
SeeAlso: AH=3Dh,AH=45h
--------D-2147------------------------------
INT 21 - DOS 2+ - "CWD" - GET CURRENT DIRECTORY
  AH = 47h
  DL = drive number (00h = default, 01h = A:, etc)
  DS:SI -> 64-byte buffer for ASCIZ pathname
Return: CF clear if successful
      AX = 0100h (undocumented)
  CF set on error
      AX = error code (0Fh) (see #01680 at AH=59h/BX=0000h)
```

```
Notes:  the returned path does not include a drive or the initial backslash
  many Microsoft products for Windows rely on AX being 0100h on success
  under the FlashTek X-32 DOS extender, the buffer pointer is in DS:ESI
SeeAlso: AH=19h,AH=3Bh,AH=71h,INT 15/AX=DE25h
--------D-2148------------------------------
INT 21 - DOS 2+ - ALLOCATE MEMORY
  AH = 48h
  BX = number of paragraphs to allocate
Return: CF clear if successful
      AX = segment of allocated block
  CF set on error
      AX = error code (07h,08h) (see #01680 at AH=59h/BX=0000h)
      BX = size of largest available block
Notes:  DOS 2.1-6.0 coalesces free blocks while scanning for a block to
    allocate
  .COM programs are initially allocated the largest available memory
    block, and should free some memory with AH=49h before attempting any
    allocations
  under the FlashTek X-32 DOS extender, EBX contains a protected-mode
    near pointer to the allocated block on a successful return
SeeAlso: AH=49h,AH=4Ah,AH=58h,AH=83h
--------D-2149------------------------------
INT 21 - DOS 2+ - FREE MEMORY
  AH = 49h
  ES = segment of block to free
Return: CF clear if successful
  CF set on error
      AX = error code (07h,09h) (see #01680 at AH=59h/BX=0000h)
Notes:  apparently never returns an error 07h, despite official docs; DOS 2.1+
    code contains only an error 09h exit
  DOS 2.1-6.0 does not coalesce adjacent free blocks when a block is
    freed, only when a block is allocated or resized
  the code for this function is identical in DOS 2.1-6.0 except for
    calls to start/end a critical section in DOS 3.0+
SeeAlso: AH=48h,AH=4Ah
--------D-214A------------------------------
INT 21 - DOS 2+ - RESIZE MEMORY BLOCK
  AH = 4Ah
  BX = new size in paragraphs
  ES = segment of block to resize
Return: CF clear if successful
```

```
  CF set on error
      AX = error code (07h,08h,09h) (see #01680 at AH=59h/BX=0000h)
      BX = maximum paragraphs available for specified memory block
Notes:  under DOS 2.1-6.0, if there is insufficient memory to expand the block
    as much as requested, the block will be made as large as possible
  DOS 2.1-6.0 coalesces any free blocks immediately following the block
    to be resized
SeeAlso: AH=48h,AH=49h,AH=83h
--------v-214A--BX00B6-----------------------
INT 21 - VIRUS???
  AH = 4Ah
  BX = 00B6h
  ES = CX
Return: ???
Note: this call is intercepted by the Search&Destroy SDRes v27.03 bundled
    with Novell DOS 7, and is presumably some virus's installation check
SeeAlso: AH=0Eh/DL=ADh,AH=4Ah/BX=FFFFh,AH=D2h"VIRUS"
--------v-214A--BXFFFF-----------------------
INT 21 - VIRUS???
  AH = 4Ah
  BX = FFFFh
  CX = 0568h
  SI = 0129h
  DI = 0000h
  ES = BP
Return: ???
Note: this call is intercepted by the Search&Destroy SDRes v27.03 bundled
    with Novell DOS 7, and is presumably some virus's installation check
SeeAlso: AH=0Eh/DL=ADh,AH=4Ah/BX=00B6h
--------D-214B--------------------------------
INT 21 - DOS 2+ - "EXEC" - LOAD AND/OR EXECUTE PROGRAM
  AH = 4Bh
  AL = type of load
      00h load and execute
      01h load but do not execute
      03h load overlay (see #01591)
      04h load and execute in background (European MS-DOS 4.0 only)
    "Exec & Go" (see also AH=80h)
  DS:DX -> ASCIZ program name (must include extension)
  ES:BX -> parameter block (see #01590,#01591,#01592)
  CX = mode (subfunction 04h only)
```

```
    0000h child placed in zombie mode after termination
    0001h child's return code discarded on termination
Return: CF clear if successful
      BX,DX destroyed
      if subfunction 01h, process ID set to new program's PSP; get with
    INT 21/AH=62h
  CF set on error
      AX = error code (01h,02h,05h,08h,0Ah,0Bh) (see #01680 at AH=59h)
Notes:  DOS 2.x destroys all registers, including SS:SP
  under ROM-based DOS, if no disk path characters (colons or slashes)
    are included in the program name, the name is searched for in the
    ROM module headers (see #01595) before searching on disk
  for functions 00h and 01h, the calling process must ensure that there
    is enough unallocated memory available; if necessary, by releasing
    memory with AH=49h or AH=4Ah
  for function 01h, the AX value to be passed to the child program is put
    on top of the child's stack
  for function 03h, DOS assumes that the overlay is being loaded into
    memory allocated by the caller
  function 01h was undocumented prior to the release of DOS 5.0
  some versions (such as DR DOS 6.0) check the parameters and parameter
    block and return an error if an invalid value (such as an offset of
    FFFFh) is found
  background programs under European MS-DOS 4.0 must use the new
    executable format
  this function ignores the filename extension, instead checking the
    first two bytes of the file to determine whether there is a valid
    .EXE header (see #01594); if not, the file is assumed to be in .COM
    format.  If present, the file may be in any of several formats which
    are extensions of the original .EXE format (see #01593)
  .COM-format executables begin running with the following register
    values:
    AL = 00h if first FCB has valid drive letter, FFh if not
    AH = 00h if second FCB has valid drive letter, FFh if not
    CS,DS,ES,SS = PSP segment
    SP = offset of last word available in first 64K segment
    (note: AX is always 0000h under DESQview)
  old-format executables begin running with the following register
    values:
    AL = 00h if first FCB has valid drive letter, FFh if not
    AH = 00h if second FCB has valid drive letter, FFh if not
```

```
    DS,ES = PSP segment
    SS:SP as defined in .EXE header
    (note: AX is always 0000h under DESQview)
  new executables begin running with the following register values
    AX = environment segment
    BX = offset of command tail in environment segment
    CX = size of automatic data segment (0000h = 64K)
    ES,BP = 0000h
    DS = automatic data segment
    SS:SP = initial stack
    the command tail corresponds to an old executable's PSP:0081h and
    following, except that the 0Dh is turned into a NUL (00h); new
    format executables have no PSP
  under the FlashTek X-32 DOS extender, only function 00h is supported
    and the pointers are passed in DS:EDX and ES:EBX
  DR DOS 6 always loads .EXE-format programs with no fixups and
    .COM-format programs starting with 9Ch 55h (PUSHF/PUSH BP) above the
    64K mark to avoid the EXEPACK bug, by extending the memory block
    containing the program's environment; this code is disabled if the
    name of the parent program as stored in the MCB is 'WIN'.
  DR DOS 3.41 and 5.0 check for a valid filename before testing the
    subfunction number, so the otherwise invalid subfunction 02h will
    only return error code 01h if the given filename actually exists;
    otherwise, errors 02h, 03h, or 05h are returned
BUGS: DOS 2.00 assumes that DS points at the current program's PSP
  Load Overlay (subfunction 03h) loads up to 512 bytes too many if the
    file contains additional data after the actual overlay
  Load but Do Not Execute (subfunction 01h) is reported to corrupt the
    top word of the caller's stack if the loaded module terminates with
    INT 21/AH=4Ch in some versions of MS-DOS, including v5.00.
SeeAlso: AX=4B05h,AH=4Ch,AH=4Dh,AH=64h/BX=0025h,AH=8Ah,INT 2E,INT 60/DI=0604h

Format of EXEC parameter block for AL=00h,01h,04h:
Offset  Size  Description (Table 01590)
 00h  WORD  segment of environment to copy for child process (copy caller's
      environment if 0000h)
 02h  DWORD pointer to command tail to be copied into child's PSP
 06h  DWORD pointer to first FCB to be copied into child's PSP
 0Ah  DWORD pointer to second FCB to be copied into child's PSP
 0Eh  DWORD (AL=01h) will hold subprogram's initial SS:SP on return
 12h  DWORD (AL=01h) will hold entry point (CS:IP) on return
```

```
SeeAlso: #01591,#01592

Format of EXEC parameter block for AL=03h:
Offset  Size  Description (Table 01591)
 00h  WORD  segment at which to load overlay
 02h  WORD  relocation factor to apply to overlay if in .EXE format
SeeAlso: #01590,#01592

Format of EXEC parameter block for FlashTek X-32:
Offset  Size  Description (Table 01592)
 00h  PWORD 48-bit far pointer to environment string
 06h  PWORD 48-bit far pointer to command tail string
SeeAlso: #01590,#01591

(Table 01593)
Values for the executable types understood by various environments:
 MZ old-style DOS executable (see #01594)
 ZM used by some very early DOS linkers, and still supported as an
    alternate to the MZ signature by MS-DOS, PC DOS, PTS-DOS, and S/DOS
 NE Windows or OS/2 1.x segmented ("new") executable (see #01596)
 LE Windows virtual device driver (VxD) linear executable (see #01609)
 LX variant of LE used in OS/2 2.x (see #01609)
 W3 Windows WIN386.EXE file; a collection of LE files
 W4 Windows95 VMM32.VXD file
 PE Win32 (Windows NT and Win32s) portable executable based on Unix COFF
 DL HP 100LX/200LX system manager compliant executable (.EXM)
 MP old PharLap .EXP (see #01619)
 P2 PharLap 286 .EXP (see #01620)
 P3 PharLap 386 .EXP (see #01620)

Format of .EXE file header:
Offset  Size  Description (Table 01594)
 00h  2 BYTEs .EXE signature, either "MZ" or "ZM" (5A4Dh or 4D5Ah)
      (see also #01593)
 02h  WORD  number of bytes in last 512-byte page of executable
 04h  WORD  total number of 512-byte pages in executable (includes any
    partial last page)
 06h  WORD  number of relocation entries
 08h  WORD  header size in paragraphs
 0Ah  WORD  minimum paragraphs of memory required to allocate in addition
      to executable's size
```

```
 0Ch  WORD  maximum paragraphs to allocate in addition to executable's size
 0Eh  WORD  initial SS relative to start of executable
 10h  WORD  initial SP
 12h  WORD  checksum (one's complement of sum of all words in executable)
 14h  DWORD initial CS:IP relative to start of executable
 18h  WORD  offset within header of relocation table
    40h or greater for new-format (NE,LE,LX,W3,PE,etc.) executable
 1Ah  WORD  overlay number (normally 0000h = main program)
---new executable---
 1Ch  4 BYTEs ???
 20h  WORD  behavior bits
 22h 26 BYTEs reserved for additional behavior info
 3Ch  DWORD offset of new executable (NE,LE,etc) header within disk file,
    or 00000000h if plain MZ executable
---Borland TLINK---
 1Ch  2 BYTEs ??? (apparently always 01h 00h)
 1Eh  BYTE  signature FBh
 1Fh  BYTE  TLINK version (major in high nybble, minor in low nybble)
 20h  2 BYTEs ??? (v2.0 apparently always 72h 6Ah, v3.0+ seems always 6Ah 72h)
---ARJ self-extracting archive---
 1Ch  4 BYTEs signature "RJSX" (older versions, new signature is "aRJsfX" in
    the first 1000 bytes of the file)
---LZEXE 0.90 compressed executable---
 1Ch  4 BYTEs signature "LZ09"
---LZEXE 0.91 compressed executable---
 1Ch  4 BYTEs signature "LZ91"
---PKLITE compressed executable---
 1Ch  BYTE  minor version number
 1Dh  BYTE  bits 0-3: major version
    bit 4: extra compression
    bit 5: huge (multi-segment) file
 1Eh  6 BYTEs signature "PKLITE" (followed by copyright message)
---LHarc 1.x self-extracting archive---
 1Ch  4 BYTEs unused???
 20h  3 BYTEs jump to start of extraction code
 23h  2 BYTEs ???
 25h 12 BYTEs signature "LHarc's SFX "
---LHA 2.x self-extracting archive---
 1Ch  8 BYTEs ???
 24h 10 BYTEs signature "LHa's SFX " (v2.10) or "LHA's SFX " (v2.13)
---TopSpeed C 3.0 CRUNCH compressed file---
```

```
 1Ch  DWORD 018A0001h
 20h  WORD  1565h
---PKARCK 3.5 self-extracting archive---
 1Ch  DWORD 00020001h
 20h  WORD  0700h
---BSA (Soviet archiver) self-extracting archive---
 1Ch  WORD  000Fh
 1Eh  BYTE  A7h
---LARC self-extracting archive---
 1Ch  4 BYTEs ???
 20h 11 BYTEs "SFX by LARC "
---LH self-extracting archive---
 1Ch  8 BYTEs ???
 24h  8 BYTEs "LH's SFX "
---RAR self-extracting archive---
 1Ch  4 BYTEs signature "RSFX"
---other linkers---
 1Ch  var optional information
---
  N   N DWORDs  relocation items
    each is the segment:offset from start of load image at which
      to add the actual load segment to the indicated WORD
Notes:  if the word at offset 02h is 4, it should be treated as 00h, since
    pre-1.10 versions of the MS linker set it that way
  if both minimum and maximum allocation (offset 0Ah/0Ch) are zero, the
    program is loaded as high in memory as possible (DOS only checks
    the maximum allocation, however)
  the maximum allocation is set to FFFFh by default
  additional data may be contained in the file beyond the end of the
    load image described by the .EXE header; this data may be overlays,
    the actual executable for newer-format executables, or debugging
    information (see #01600,#01624)
  relocations entries need not be in any particular order, although they
    are typically stored in order from beginning to end of the load
    image
SeeAlso: #01596

Format of ROM Module Header:
Offset  Size  Description (Table 01595)
 00h  2 BYTEs ROM signature 55h, AAh
 02h  BYTE  size of ROM in 512-byte blocks
```

```
 03h  3 BYTEs POST initialization entry point (near JMP instruction)
 06h  ROM Program Name List [array]
  Offset  Size  Description
   00h  BYTE  length of ROM program's name (00h if end of name list)
   01h  N BYTEs program name
   N+1  3 BYTEs program entry point (near JMP instruction)

Format of new executable header:
Offset  Size  Description (Table 01596)
 00h  2 BYTEs "NE" (4Eh 45h) signature
 02h  2 BYTEs linker version (major, then minor)
 04h  WORD  offset from start of this header to entry table (see #01603)
 06h  WORD  length of entry table in bytes
 08h  DWORD file load CRC (0 in Borland's TPW)
 0Ch  BYTE  program flags (see #01597)
 0Dh  BYTE  application flags (see #01598)
 0Eh  WORD  auto data segment index
 10h  WORD  initial local heap size
 12h  WORD  initial stack size (added to data seg, 0000h if SS <> DS)
 14h  DWORD program entry point (CS:IP), "CS" is index into segment table
 18h  DWORD initial stack pointer (SS:SP), "SS" is segment index
    if SS=automatic data segment and SP=0000h, the stack pointer is
      set to the top of the automatic data segment, just below the
      local heap
 1Ch  WORD  segment count
 1Eh  WORD  module reference count
 20h  WORD  length of nonresident names table in bytes
 22h  WORD  offset from start of this header to segment table (see #01601)
 24h  WORD  offset from start of this header to resource table
 26h  WORD  offset from start of this header to resident names table
 28h  WORD  offset from start of this header to module reference table
 2Ah  WORD  offset from start of this header to imported names table
    (array of counted strings, terminated with a string of length
      00h)
 2Ch  DWORD offset from start of file to nonresident names table
 30h  WORD  count of moveable entry point listed in entry table
 32h  WORD  file alignment size shift count
    0 is equivalent to 9 (default 512-byte pages)
 34h  WORD  number of resource table entries
 36h  BYTE  target operating system
    00h unknown
```

```
    01h OS/2
    02h Windows
    03h European MS-DOS 4.x
    04h Windows 386
    05h BOSS (Borland Operating System Services)
    81h PharLap 286|DOS-Extender, OS/2
    82h PharLap 286|DOS-Extender, Windows
 37h  BYTE  other EXE flags (see #01599)
 38h  WORD  offset to return thunks or start of gangload area
 3Ah  WORD  offset to segment reference thunks or length of gangload area
 3Ch  WORD  minimum code swap area size
 3Eh  2 BYTEs expected Windows version (minor version first)
Note: this header is documented in detail in the Windows 3.1 SDK Programmer's
    Reference, Vol 4.
SeeAlso: #01594

Bitfields for new executable program flags:
Bit(s)  Description (Table 01597)
 0-1  DGROUP type
    0 = none
    1 = single shared
    2 = multiple (unshared)
    3 = (null)
 2  global initialization
 3  protected mode only
 4  8086 instructions
 5  80286 instructions
 6  80386 instructions
 7  80x87 instructions

Bitfields for new executable application flags:
Bit(s)  Description (Table 01598)
 0-2  application type
  001 full screen (not aware of Windows/P.M. API)
  010 compatible with Windows/P.M. API
  011 uses Windows/P.M. API
 3  is a Family Application (OS/2)
 5  0=executable, 1=errors in image
 6  non-conforming program (valid stack is not maintained)
 7  DLL or driver rather than application
  (SS:SP info invalid, CS:IP points at FAR init routine called with
```

```
    AX=module handle which returns AX=0000h on failure, AX nonzero on
    successful initialization)

Bitfields for other new .EXE flags:
Bit(s)  Description (Table 01599)
 0  supports long filenames
 1  2.X protected mode
 2  2.X proportional font
 3  gangload area

Format of Codeview trailer (at end of executable):
Offset  Size  Description (Table 01600)
 00h  WORD  signature 4E42h ('NB')
 02h  WORD  Microsoft debug info version number
 04h  DWORD Codeview header offset
SeeAlso: #01624

Format of new executable segment table record:
Offset  Size  Description (Table 01601)
 00h  WORD  offset in file (shift left by alignment shift to get byte offs)
 02h  WORD  length of image in file (0000h = 64K)
 04h  WORD  segment attributes (see #01602)
 06h  WORD  number of bytes to allocate for segment (0000h = 64K)
Note: the first segment table entry is entry number 1
SeeAlso: #01604

Bitfields for segment attributes:
Bit(s)  Description (Table 01602)
 0  data segment rather than code segment
 1  unused???
 2  real mode
 3  iterated
 4  movable
 5  sharable
 6  preloaded rather than demand-loaded
 7  execute-only (code) or read-only (data)
 8  relocations (directly following code for this segment)
 9  debug info present
 10,11  80286 DPL bits
 12 discardable
 13-15  discard priority
```

```
Format of new executable entry table item (list):
Offset  Size  Description (Table 01603)
 00h  BYTE  number of entry points (00h if end of entry table list)
 01h  BYTE  segment number (00h if end of entry table list)
 02h 3N BYTEs entry records
    Offset  Size  Description
     00h  BYTE  flags
        bit 0: exported
        bit 1: single data
        bits 2-7: unused???
     01h  WORD  offset within segment

Format of new executable relocation data (immediately follows segment image):
Offset  Size  Description (Table 01604)
 00h  WORD  number of relocation items
 02h 8N BYTEs relocation items
    Offset  Size  Description
     00h  BYTE  relocation type
        00h LOBYTE
        02h BASE
        03h PTR
        05h OFFS
        0Bh PTR48
        0Dh OFFS32
     01h  BYTE  flags
        bit 2: additive
     02h  WORD  offset within segment
     04h  WORD  target address segment
     06h  WORD  target address offset
SeeAlso: #01601,#01605

Format of new executable resource data:
Offset  Size  Description (Table 01605)
 00h  WORD  alignment shift count for resource data
 02h  N RECORDs resources
  Format of resource record:
  Offset  Size  Description
   00h  WORD  type ID
      0000h if end of resource records
      >= 8000h if integer type
```

```
      else offset from start of resource table to type string
   02h  WORD  number of resources of this type
   04h  DWORD reserved for runtime use
   08h  N Resources (see #01606)
Note: resource type and name strings are stored immediately following the
    resource table, and are not null-terminated
SeeAlso: #01606

Format of new executable resource entry:
Offset  Size  Description (Table 01606)
 00h  WORD  offset in alignment units from start of file to contents of
    the resource data
 02h  WORD  length of resource image in bytes
 04h  WORD  flags
    bit 4: moveable
    bit 5: shareable
    bit 6: preloaded
 06h  WORD  resource ID
    >= 8000h if integer resource
    else offset from start of resource table to resource string
 08h  DWORD reserved for runtime use
Notes:  resource type and name strings are stored immediately following the
    resource table, and are not null-terminated
  strings are counted strings, with a string of length 0 indicating the
    end of the resource table
SeeAlso: #01605,#01607

Format of new executable module reference table [one bundle of entries]:
Offset  Size  Description (Table 01607)
 00h  BYTE  number of records in this bundle (00h if end of table)
 01h  BYTE  segment indicator
    00h unused
    FFh movable segment, segment number is in entry
    else segment number of fixed segment
 02h  N RECORDs
  Format of segment record
  Offset  Size  Description
   00h  BYTE  flags
      bit 0: entry is exported
      bit 1: entry uses global (shared) data
      bits 7-3: number of parameter words
```

```
  ---fixed segment---
   01h  WORD  offset
  ---moveable segment---
   01h  2 BYTEs INT 3F instruction (CDh 3Fh)
   03h  BYTE  segment number
   05h  WORD  offset
Note: table entries are numbered starting from 1
SeeAlso: #01608

Format of new executable resident/nonresident name table entry:
Offset  Size  Description (Table 01608)
 00h  BYTE  length of string (00h if end of table)
 01h  N BYTEs ASCII text of string
 N+1  WORD  ordinal number (index into entry table)
Notes:  the first string in the resident name table is the module name; the
    first entry in the nonresident name table is the module description
  the strings are case-sensitive; if the executable was linked with
    /IGNORECASE, all strings are in uppercase
SeeAlso: #01607

Format of Linear Executable (enhanced mode executable) header:
Offset  Size  Description (Table 01609)
 00h  2 BYTEs "LE" (4Ch 45h) signature (Windows)
    "LX" (4Ch 58h) signature (OS/2)
 02h  BYTE  byte order (00h = little-endian, nonzero = big-endian)
 03h  BYTE  word order (00h = little-endian, nonzero = big-endian)
 04h  DWORD executable format level
 08h  WORD  CPU type (see also INT 15/AH=C9h)
    01h Intel 80286 or upwardly compatible
    02h Intel 80386 or upwardly compatible
    03h Intel 80486 or upwardly compatible
    04h Intel Pentium (80586) or upwardly compatible
    20h Intel i860 (N10) or compatible
    21h Intel "N11" or compatible
    40h MIPS Mark I (R2000, R3000) or compatible
    41h MIPS Mark II (R6000) or compatible
    42h MIPS Mark III (R4000) or compatible
 0Ah  WORD  target operating system
    01h OS/2
    02h Windows
    03h European DOS 4.0
```

```
   04h Windows 386
0Ch  DWORD module version
10h  DWORD module type (see #01610)
14h  DWORD number of memory pages
18h  Initial CS:EIP
 DWORD object number
 DWORD offset
20h  Initial SS:ESP
 DWORD object number
 DWORD offset
28h  DWORD memory page size
2Ch  DWORD (Windows LE) bytes on last page
   (OS/2 LX) page offset shift count
30h  DWORD fixup section size
34h  DWORD fixup section checksum
38h  DWORD loader section size
3Ch  DWORD loader section checksum
40h  DWORD offset of object table (see #01611)
44h  DWORD object table entries
48h  DWORD object page map table offset (see #01613)
4Ch  DWORD object iterate data map offset
50h  DWORD resource table offset
54h  DWORD resource table entries
58h  DWORD resident names table offset (see #01614)
5Ch  DWORD entry table offset (see #01615,#01616)
60h  DWORD module directives table offset
64h  DWORD Module Directives entries
68h  DWORD Fixup page table offset
6Ch  DWORD Fixup record table offset (see #01618)
70h  DWORD imported modules name table offset
74h  DWORD imported modules count
78h  DWORD imported procedures name table offset
7Ch  DWORD per-page checksum table offset
80h  DWORD data pages offset
84h  DWORD preload page count
88h  DWORD non-resident names table offset
8Ch  DWORD non-resident names table length
90h  DWORD non-resident names checksum
94h  DWORD automatic data object
98h  DWORD debug information offset
9Ch  DWORD debug information length
```

```
 A0h  DWORD preload instance pages number
 A4h  DWORD demand instance pages number
 A8h  DWORD extra heap allocation
 ACh 12 BYTEs reserved
 B8h  DWORD offset of VERSIONINFO resource (MS-Windows VxD only)
 BCh  DWORD pointer to ??? (dynamically-loadable VxDs only???)
 C0h  WORD  device ID (MS-Windows VxD only)
 C2h  WORD  DDK version (MS-Windows VxD only)
Note: used by EMM386.EXE, QEMM, and Windows 3.0 Enhanced Mode drivers

Bitfields for Linear Executable module type:
Bit(s)  Description (Table 01610)
 2  initialization (only for DLLs) 0 = global, 1 = per-process
 4  no internal fixups in executable image
 5  no external fixups in executable image
 8-10 API compatibility
  0 = unknown
  1 = incompatible with PM windowing \
  2 = compatible with PM windowing    > (only for
  3 = uses PM windowing API    /  programs)
 13 module not loadable (only for programs)
 15-17  module type
  000 program
  001 library (DLL)
  011 protected memory library module
  100 physical device driver
  110 virtual device driver
 30 per-process library termination
  (requires valid CS:EIP, can't be set for .EXE)

Format of object table entry:
Offset  Size  Description (Table 01611)
 00h  DWORD virtual size in bytes
 04h  DWORD relocation base address
 08h  DWORD object flags (see #01612)
 0Ch  DWORD page map index
 10h  DWORD page map entries (see #01613)
 14h  4 BYTEs reserved??? (apparently always zeros)

Bitfields for object flags:
Bit(s)  Description (Table 01612)
```

```
 0  readable
 1  writable
 2  executable
 3  resource
 4  discardable
 5  shared
 6  preloaded
 7  invalid
 8-9  type
  00 normal
  01 zero-filled
  10 resident
  11 resident and contiguous
 10 resident and long-lockable
 11 reserved
 12 16:16 alias required
 13 "BIG" (Huge: 32-bit)
 14 conforming
 15 "OBJECT_I/O_PRIVILEGE_LEVEL"
 16-31  reserved

Format of object page map table entry:
Offset  Size  Description (Table 01613)
 00h  BYTE  ??? (usually 00h)
 01h  WORD  (big-endian) index to fixup table
    0000h if no relocation info
 03h  BYTE  type (00h hard copy in file, 03h some relocation needed)

Format of resident names table entry:
Offset  Size  Description (Table 01614)
 00h  BYTE  length of name
 01h  N BYTEs name
 N+1  3 BYTEs ???

Format of LE linear executable entry table:
Offset  Size  Description (Table 01615)
 00h  BYTE  number of entries in table
 01h 10 BYTEs per entry
    Offset  Size  Description
     00h  BYTE  bit flags
        bit 0: non-empty bundle
```

```
        bit 1: 32-bit entry
     01h  WORD  object number
     03h  BYTE  entry type flags
        bit 0: exported
        bit 1: uses single data rather than instance
        bit 2: reserved
        bits 3-7: number of stack parameters
     04h  DWORD offset of entry point
     08h  2 BYTEs ???
Note: empty bundles (bit flags at 00h = 00h) are used to skip unused indices,
    and do not contain the remaining nine bytes

Format of LX linear executable entry table [array]:
Offset  Size  Description (Table 01616)
 00h  BYTE  number of bundles following (00h = end of entry table)
 01h  BYTE  bundle type
    00h empty
    01h 16-bit entry
    02h 286 callgate entry
    03h 32-bit entry
    04h forwarder entry
    bit 7 set if additional parameter typing information is present
---bundle type 00h---
 no additional fields
---bundle type 01h---
 02h  WORD  object number
 04h  BYTE  entry flags
    bit 0: exported
    bits 7-3: number of stack parameters
 05h  WORD  offset of entry point in object (shifted by page size shift)
---bundle type 02h---
 02h  WORD  object number
 04h  BYTE  entry flags
    bit 0: exported
    bits 7-3: number of stack parameters
 05h  WORD  offset of entry point in object
 07h  WORD  reserved for callgate selector (used by loader)
---bundle type 03h---
 02h  WORD  object number
 04h  BYTE  entry flags
    bit 0: exported
```

```
    bits 7-3: number of stack parameters
 05h  DWORD offset of entry point in object
---bundle type 04h---
 02h  WORD  reserved
 04h  BYTE  forwarder flags
    bit 0: import by ordinal
    bits 7-1 reserved
 05h  WORD  module ordinal
    (forwarder's index into Import Module Name table)
 07h  DWORD procedure name offset or import ordinal number
Note: all fields after the first two bytes are repeated N times

Bitfields for linear executable fixup type:
Bit(s)  Description (Table 01617)
 7  ordinal is BYTE rather than WORD
 6  16-rather than 8-object number/module ordinal
 5  addition with DWORD rather than WORD
 4  relocation info has size with new two bytes at end
 3  reserved (0)
 2  set if add to destination, clear to replace destination
 1-0  type
  00 internal fixup
  01 external fixup, imported by ordinal
  10 external fixup, imported by name
  11 internal fixup via entry table

Format of linear executable fixup record:
Offset  Size  Description (Table 01618)
 00h  BYTE  type
    bits 7-4: modifier (0001 single, 0011 multiple)
    bits 3-0: type
      0000 byte offset
      0010 word segment
      0011 16-bit far pointer (DWORD)
      0101 16-bit offset
      0110 32-bit far pointer (PWORD)
      0111 32-bit offset
      1000 near call or jump, WORD/DWORD based on seg attrib
 01h  BYTE  linear executable fixup type (see #01617)
---if single type---
 02h  WORD  offset within page
```

```
 04h  relocation information
  ---internal fixup---
  BYTE  object number
  ---external,ordinal---
  BYTE  one-based module number in Import Module table
  BYTE/WORD ordinal number
  WORD/DWORD value to add (only present if modifier bit 4 set)
  ---external,name---
  BYTE  one-based module number in Import Module table
  WORD  offset in Import Procedure names
  WORD/DWORD value to add (only present if modifier bit 4 set)
---if multiple type---
 02h  BYTE  number of items
 03h  var relocation info as for "single" type (above)
      N WORDs offsets of items to relocate

Format of old Phar Lap .EXP file header:
Offset  Size  Description (Table 01619)
 00h  2 BYTEs "MP" (4Dh 50h) signature
 02h  WORD  remainder of image size / page size (page size = 512h)
 04h  WORD  size of image in pages
 06h  WORD  number of relocation items
 08h  WORD  header size in paragraphs
 0Ah  WORD  minimum number of extra 4K pages to be allocated at the end
      of program, when it is loaded
 0Ch  WORD  maximum number of extra 4K pages to be allocated at the end
      of program, when it is loaded
 0Eh  DWORD initial ESP
 12h  WORD  word checksum of file
 14h  DWORD initial EIP
 18h  WORD  offset of first relocation item
 1Ah  WORD  overlay number
 1Ch  WORD  ??? (wants to be 1)
SeeAlso: #01620

Format of new Phar Lap .EXP file header:
Offset  Size  Description (Table 01620)
 00h  2 BYTEs signature ("P2" for 286 .EXP executable, "P3" for 386 .EXP)
 02h  WORD  level (01h flat-model file, 02h multisegmented file)
 04h  WORD  header size
 06h  DWORD file size in bytes
```

```
 0Ah  WORD  checksum
 0Ch  DWORD offset of run-time parameters within file (see #01622)
 10h  DWORD size of run-time parameters in bytes
 14h  DWORD offset of relocation table within file
 18h  DWORD size of relocation table in bytes
 1Ch  DWORD offset of segment information table within file (see #01621)
 20h  DWORD size of segment information table in bytes
 24h  WORD  size of segment information table entry in bytes
 26h  DWORD offset of load image within file
 2Ah  DWORD size of load image on disk
 2Eh  DWORD offset of symbol table within file or 00000000h
 32h  DWORD size of symbol table in bytes
 36h  DWORD offset of GDT within load image
 3Ah  DWORD size of GDT in bytes
 3Eh  DWORD offset of LDT within load image
 42h  DWORD size of LDT in bytes
 46h  DWORD offset of IDT within load image
 4Ah  DWORD size of IDT in bytes
 4Eh  DWORD offset of TSS within load image
 52h  DWORD size of TSS in bytes
 56h  DWORD minimum number of extra bytes to be allocated at end of program
    (level 1 executables only)
 5Ah  DWORD maximum number of extra bytes to be allocated at end of program
    (level 1 executables only)
 5Eh  DWORD base load offset (level 1 executables only)
 62h  DWORD initial ESP
 66h  WORD  initial SS
 68h  DWORD initial EIP
 6Ch  WORD  initial CS
 6Eh  WORD  initial LDT
 70h  WORD  initial TSS
 72h  WORD  flags
    bit 0: load image is packed
    bit 1: 32-bit checksum is present
    bits 4-2: type of relocation table
 74h  DWORD memory requirements for load image
 78h  DWORD 32-bit checksum (optional)
 7Ch  DWORD size of stack segment in bytes
 80h 256 BYTEs  reserved (0)
SeeAlso: #01619,#01623
```

```
Format of Phar Lap segment information table entry:
Offset  Size  Description (Table 01621)
 00h  WORD  selector number
 02h  WORD  flags
 04h  DWORD base offset of selector
 08h  DWORD minimum number of extra bytes to be allocated to the segment

Format of 386|DOS-Extender run-time parameters:
Offset  Size  Description (Table 01622)
 00h  2 BYTEs signature "DX" (44h 58h)
 02h  WORD  minimum number of real-mode params to leave free at run time
 04h  WORD  maximum number of real-mode params to leave free at run time
 06h  WORD  minimum interrupt buffer size in KB
 08h  WORD  maximum interrupt buffer size in KB
 0Ah  WORD  number of interrupt stacks
 0Ch  WORD  size in KB of each interrupt stack
 0Eh  DWORD offset of byte past end of real-mode code and data
 12h  WORD  size in KB of call buffers
 14h  WORD  flags
    bit 0: file is virtual memory manager
    bit 1: file is a debugger
 16h  WORD  unprivileged flag (if nonzero, executes at ring 1, 2, or 3)
 18h 104 BYTEs  reserved (0)

Format of Phar Lap repeat block header:
Offset  Size  Description (Table 01623)
 00h  WORD  byte count
 02h  BYTE  repeat string length

Format of Borland debugging information header (following load image):
Offset  Size  Description (Table 01624)
 00h  WORD  signature 52FBh
 02h  WORD  version ID
 04h  DWORD size of name pool in bytes
 08h  WORD  number of names in name pool
 0Ah  WORD  number of type entries
 0Ch  WORD  number of structure members
 0Eh  WORD  number of symbols
 10h  WORD  number of global symbols
 12h  WORD  number of modules
 14h  WORD  number of locals (optional)
```

```
 16h  WORD  number of scopes in table
 18h  WORD  number of line-number entries
 1Ah  WORD  number of include files
 1Ch  WORD  number of segment records
 1Eh  WORD  number of segment/file correlations
 20h  DWORD size of load image after removing uninitialized data and debug
      information
 24h  DWORD debugger hook; pointer into debugged program whose meaning
      depends on program flags
 28h  BYTE  program flags
    bit 0: case-sensitive link
    bit 1: pascal overlay program
 29h  WORD  no longer used
 2Bh  WORD  size of data pool in bytes
 2Dh  BYTE  padding
 2Eh  WORD  size of following header extension (currently 00h, 10h, or 20h)
 30h  WORD  number of classes
 32h  WORD  number of parents
 34h  WORD  number of global classes (currently unused)
 36h  WORD  number of overloads (currently unused)
 38h  WORD  number of scope classes
 3Ah  WORD  number of module classes
 3Ch  WORD  number of coverage offsets
 3Eh  DWORD offset relative to symbol base of name pool
 42h  WORD  number of browser information records
 44h  WORD  number of optimized symbol records
 46h  WORD  debugging flags
 48h  8 BYTEs padding
Note: additional information on the Borland debugging info may be found in
    Borland's Open Architecture Handbook
SeeAlso: #01600
--------U-214B--------------------------------
INT 21 - ELRES v1.0 only - INSTALLATION CHECK
  AH = 4Bh
  DS:DX = 0000h:0000h
Return: ES:BX -> ELRES history structure (see #01381 at AH=2Bh/CX=454Ch)
  DX = DABEh (signature, DAve BEnnett)
Program: ELRES is an MS-DOS return code (errorlevel) recorder by David H.
    Bennett
SeeAlso: AH=2Bh/CX=454Ch
--------v-214B04------------------------------
```

```
INT 21 - VIRUS - "MG", "699"/"Thirteen Minutes" - INSTALLATION CHECK
  AX = 4B04h
Return: CF clear if "MG" resident
  AX = 044Bh if "699"/"Thirteen Minutes" resident
SeeAlso: AX=4243h,AH=4Ah/BX=FFFFh,AX=4B21h
--------D-214B05----------------------------
INT 21 - DOS 5+ - SET EXECUTION STATE
  AX = 4B05h
  DS:DX -> execution state structure (see #01625)
Return: CF clear if successful
      AX = 0000h
  CF set on error
      AX = error code (see #01680 at AH=59h/BX=0000h)
Note: used by programs which intercept AX=4B00h to prepare new programs for
    execution (including setting the DOS version number).  No DOS, BIOS
    or other software interrupt may be called after return from this call
    before commencement of the child process.  If DOS is running in the
    HMA, A20 is turned off on return from this call.
SeeAlso: AH=4Bh

Format of execution state structure:
Offset  Size  Description (Table 01625)
 00h  WORD  reserved (00h)
 02h  WORD  type flags
    bit 0: program is an .EXE
    bit 1: program is an overlay
 04h  DWORD pointer to ASCIZ name of program file
 08h  WORD  PSP segment of new program
 0Ah  DWORD starting CS:IP of new program
 0Eh  DWORD program size including PSP
----------214B18DX0010----------------------
INT 21 U - FBOOT v2.13 - PERFORM FAST BOOTSTRAP
  AX = 4B18h
  DX = 0010h
  BX = disk selector
      0000h boot from floppy
      0080h boot from hard disk
Return: never if FastBoot installed
Program: CyberWare FastBoot allows fast warm boots by skipping CMOS
    checking, ROM scan, RAM & peripheral components test
InstallCheck: search for a character device driver called "FBOOT$$$"
```

```
--------v-214B20-----------------------------
INT 21 - VIRUS - "Holocaust"/"Telefonica" - ???
  AX = 4B20h
SeeAlso: AX=4B04h,AX=4B21h
--------v-214B21-----------------------------
INT 21 C - VIRUS - "Holocaust"/"Telefonica" - ???
  AX = 4B21h
Note: called at completion of virus installation
SeeAlso: AX=4B04h,AX=4B20h,AX=4B25h
--------v-214B25-----------------------------
INT 21 - VIRUS - "1063"/"Mono" - INSTALLATION CHECK
  AX = 4B25h
Return: DI = 1234h if resident
SeeAlso: AX=4B21h,AX=4B40h
--------v-214B40-----------------------------
INT 21 - VIRUS - "Plastique"/"AntiCad" - INSTALLATION CHECK
  AX = 4B40h
Return: AX = 5678h if resident
SeeAlso: AX=4B25h,AX=4B41h,AX=4B4Ah
--------v-214B41-----------------------------
INT 21 - VIRUS - "Plastique"/"AntiCad" - ???
  AX = 4B41h
  ???
Return: ???
SeeAlso: AX=4B40h
--------v-214B4A-----------------------------
INT 21 - VIRUS - "Jabberwocky" - INSTALLATION CHECK
  AX = 4B4Ah
Return: AL = 57h if resident
SeeAlso: AX=4B40h,AX=4B4Bh
--------v-214B4B-----------------------------
INT 21 - VIRUS - "Horse-2" - INSTALLATION CHECK
  AX = 4B4Bh
Return: CF clear if resident
SeeAlso: AX=4B4Ah,AX=4B4Dh
--------v-214B4D-----------------------------
INT 21 - VIRUS - "Murphy-2", "Patricia"/"Smack" - INSTALLATION CHECK
  AX = 4B4Dh
Return: CF clear if resident
SeeAlso: AX=4B4Ah,AX=4B50h
--------v-214B50-----------------------------
```

```
INT 21 - VIRUS - "Plastique-2576"/"AntiCad-2576" - INSTALLATION CHECK
  AX = 4B50h
Return: AX = 1234h if resident
SeeAlso: AX=4B4Dh,AX=4B53h,AX=4B60h
--------v-214B53----------------------------
INT 21 - VIRUS - "Horse" - INSTALLATION CHECK
  AX = 4B53h
Return: CF clear if resident
SeeAlso: AX=4B50h,AX=4B53h/BX=2121h,AX=4B55h
--------v-214B53BX2121----------------------
INT 21 - VIRUS - "One Half" - INSTALLATION CHECK
  AX = 4B53h
  BX = 2121h
  CX = 1212h
  DX = 0236h
Return: AX = 454Bh if installed
SeeAlso: AX=4B50h,AX=4B53h,AX=4B55h
--------v-214B55----------------------------
INT 21 - VIRUS - "Sparse" - INSTALLATION CHECK
  AX = 4B55h
Return: AX = 1231h if resident
SeeAlso: AX=4B53h,AX=4B59h
--------v-214B59----------------------------
INT 21 - VIRUS - "Murphy-1", "Murphy-4" - INSTALLATION CHECK
  AX = 4B59h
Return: CF clear if resident
SeeAlso: AX=4B50h,AX=4B5Eh
--------v-214B5E----------------------------
INT 21 - VIRUS - "Brothers" - INSTALLATION CHECK
  AX = 4B5Eh
Return: CF clear if resident
SeeAlso: AX=4B59h,AX=4B87h
--------v-214B60----------------------------
INT 21 - VIRUS - "Plastique-2576"/"AntiCad-2576" - ???
  AX = 4B60h
  ???
Return: ???
SeeAlso: AX=4B50h
--------O-214B80----------------------------
INT 21 - DR DOS v3.31+ - RUN ALREADY-LOADED KERNEL FILE
  AX = 4B80h
```

```
  DS:DX -> ASCIZ name of program to EXEC
  ES = segment of PSP for kernel file
Return: only if call failed
Note: DR DOS uses this call after an AX=4B01h to load the kernel file into
    memory and patch the program's parent-PSP field to point at itself
SeeAlso: AH=4Bh"EXEC"
--------v-214B87----------------------------
INT 21 - VIRUS - "Shirley" - INSTALLATION CHECK
  AX = 4B87h
Return: AX = 6663h if resident
SeeAlso: AX=4B5Eh,AX=4B95h
--------v-214B95----------------------------
INT 21 - VIRUS - "Zherkov-1882" - INSTALLATION CHECK
  AX = 4B95h
Return: AX = 1973h if resident
SeeAlso: AX=4B87h,AX=4BA7h
--------v-214BA7----------------------------
INT 21 - VIRUS - "1876"/"Dash-em" - INSTALLATION CHECK
  AX = 4BA7h
Return: AX = B459h if resident
SeeAlso: AX=4B95h,AX=4BAAh
--------v-214BAA----------------------------
INT 21 - VIRUS - "Nomenklatura" - INSTALLATION CHECK
  AX = 4BAAh
Return: CF clear if resident
SeeAlso: AX=4BA7h,AX=4BAFh
--------v-214BAF----------------------------
INT 21 - VIRUS - "948"/"Screenplus1", "Magnitogorsk" - INSTALLATION CHECK
  AX = 4BAFh
Return: AL = AFh if "Magnitogorsk" resident
  AL = FAh if "948"/"Screenplus1" resident
SeeAlso: AX=4BAAh,AX=4BB1h"VIRUS"
--------v-214BB1----------------------------
INT 21 - VIRUS - "UZZY" -INSTALLATION CHECK
  AX = 4BB1h
Return: CL = 04h if resident
SeeAlso: AX=4BAFh"VIRUS",AX=4BDDh"VIRUS"
--------v-214BDD----------------------------
INT 21 - VIRUS - "Lozinsky"/"Zherkov" - INSTALLATION CHECK
  AX = 4BDDh
Return: AX = 1234h
```

```
SeeAlso: AX=4BB1h"VIRUS",AX=4BFEh
--------v-214BEE----------------------------
INT 21 - F-DRIVER.SYS v1.14+ - GRAB INT 21
  AX = 4BEEh
Return: AX = status
      1234h grab was successful
      2345h failed (INT 21 grabbed previously)
Program: F-DRIVER.SYS is part of the shareware F-PROT virus/trojan protection
    package by Fridrik Skulason
Note: when called the first time, this function moves the INT 21 monitoring
    code from its original location in the INT 21 chain to be the first
    thing called by INT 21.  This is the mechanism used by F-NET.
SeeAlso: INT 2F/AX=4653h/CX=0002h,INT 2F/AX=4653h/CX=0007h
--------k-214BF0----------------------------
INT 21 - DIET v1.10+ (Overlay Mode) - INSTALLATION CHECK
  AX = 4BF0h
Return: CF clear if installed
      AX = 899Dh
Program: DIET is an executable-compression program by Teddy Matsumoto
SeeAlso: AX=37D0h,AX=4BF1h
--------k-214BF1----------------------------
INT 21 - DIET v1.10+ (Overlay Mode) - EXPAND PROGRAM???
  AX = 4BF1h
Return: ???
SeeAlso: AX=37D0h,AX=4BF0h
--------v-214BF1----------------------------
INT 21 - VIRUS - "Jerusalem 2" - INSTALLATION CHECK
  AX = 4BF1h
Return: AX = 1FB4h if resident
SeeAlso: AH=3Fh/BX=FEB0h"VIRUS",AX=4BDDh"VIRUS",AX=4BFEh"VIRUS"
--------v-214BFE----------------------------
INT 21 - VIRUS - "Hitchcock", "Dark Avenger-1028", "1193" - INSTALLATION CHECK
  AX = 4BFEh
Return: AX = 1234h if "Hitchcock" or "Storm" resident
  AX = ABCDh if "1193"/"Copyright" resident
  DI = 55BBh if "Dark Avenger-1028" resident
SeeAlso: AX=4BDDh,AX=4BF1h"VIRUS",AX=4BFFh"Justice"
--------v-214BFF----------------------------
INT 21 - VIRUS - "USSR-707", "Justice", "Europe 92" - INSTALLATION CHECK
  AX = 4BFFh
Return: BL = FFh if "USSR-707" resident
```

```
  DI = 55AAh if "Justice" resident
  AX = 1234h if "Hitchcock.1238" resident
  CF clear if "Europe 92" resident
SeeAlso: AX=4BFEh,AX=4BFFh"Cascade",AX=5252h
--------v-214BFFSI0000-----------------------
INT 21 - VIRUS - "Cascade" - INSTALLATION CHECK
  AX = 4BFFh
  SI = 0000h
  DI = 0000h
Return: DI = 55AAh if installed
SeeAlso: AX=4BFFh"Justice",AX=5252h
--------D-214C------------------------------
INT 21 - DOS 2+ - "EXIT" - TERMINATE WITH RETURN CODE
  AH = 4Ch
  AL = return code
Return: never returns
Notes:  unless the process is its own parent
    (see #01378 [offset 16h] at AH=26h), all open files are closed and
    all memory belonging to the process is freed
  all network file locks should be removed before calling this function
SeeAlso: AH=00h,AH=26h,AH=4Bh,AH=4Dh,INT 15/AH=12h/BH=02h,INT 20,INT 22
SeeAlso: INT 60/DI=0601h
--------m-214C57----------------------------
INT 21 - Headroom - ???
  AX = 4C57h
  DS:DX -> target address
Note: jumps to target address instead of terminating program
SeeAlso: AX=5758h
--------D-214D------------------------------
INT 21 - DOS 2+ - GET RETURN CODE (ERRORLEVEL)
  AH = 4Dh
Return: AH = termination type
      00h normal (INT 20,INT 21/AH=00h, or INT 21/AH=4Ch)
      01h control-C abort
      02h critical error abort
      03h terminate and stay resident (INT 21/AH=31h or INT 27)
  AL = return code
  CF clear
Notes:  the word in which DOS stores the return code is cleared after being
    read by this function, so the return code can only be retrieved once
  COMMAND.COM stores the return code of the last external command it
```

```
    executed as ERRORLEVEL
  this call should not be used if the child was started with AX=4B04h;
    use AH=8Ah instead
  the following sequence will close a Virtual DOS Machine under OS/2 2.0
    through OS/2 Merlin (but may change in the future):
      MOV AH,4Dh
      INT 21h
      HLT
      DB  02h,0FDh
    This sequence is the only way to close a specific VDM which was
    booted from floppy or a disk image.
SeeAlso: AH=4Bh,AH=4Ch,AH=8Ah
--------D-214E-----------------------------
INT 21 - DOS 2+ - "FINDFIRST" - FIND FIRST MATCHING FILE
  AH = 4Eh
  AL = special flag for use by APPEND (refer to note below)
  CX = file attribute mask (see #01420 at AX=4301h) (bits 0 and 5 ignored)
      0088h (Novell DOS 7) find first deleted file
  DS:DX -> ASCIZ file specification (may include path and wildcards)
Return: CF clear if successful
      Disk Transfer Area filled with FindFirst data block (see #01626)
  CF set on error
      AX = error code (02h,03h,12h) (see #01680 at AH=59h/BX=0000h)
Notes:  for search attributes other than 08h, all files with at MOST the
    specified combination of hidden, system, and directory attributes
    will be returned.  Under DOS 2.x, searching for attribute 08h
    (volume label) will also return normal files, while under DOS 3.0+
    only the volume label (if any) will be returned.
  this call also returns successfully if given the name of a character
    device without wildcards.  DOS 2.x returns attribute 00h, size 0,
    and the current date and time.  DOS 3.0+ returns attribute 40h and
    the current date and time.
  immediately after an INT 2F/AX=B711h (APPEND return found name), the
    name at DS:DX will be overwritten; if AL=00h on entry, the actual
    found pathname will be stored, otherwise, the actual found path
    will be prepended to the original filespec without a path.
  under LANtastic, this call may be used to obtain a list of a server's
    shared resources by searching for "\\SERVER\*.*"; a list of printer
    resources may be obtained by searching for "\\SERVER\@*.*"
  under the FlashTek X-32 DOS extender, the filespec pointer is in DS:EDX
BUGS: under DOS 3.x and 4.x, the second and subsequent calls to this function
```

```
    with a character device name (no wildcards) and search attributes
    which include the volume-label bit (08h) will fail unless there is
    an intervening DOS call which implicitly or explicity performs a
    directory search without the volume-label bit.  Such implicit
    searches are performed by CREATE (AH=3Ch), OPEN (AH=3Dh), UNLINK
    (AH=41h), and RENAME (AH=56h)
  DR DOS 3.41 and 5.0 return the Directory attribute for the volume label
SeeAlso: AH=11h,AH=4Fh,AX=4301h,AX=714Eh,AX=71A1h,AX=F257h/SF=02h
SeeAlso: INT 2F/AX=111Bh,INT 2F/AX=B711h

Format of FindFirst data block:
Offset  Size  Description (Table 01626)
---PC-DOS 3.10, PC-DOS 4.01, MS-DOS 3.2/3.3/5.0---
 00h  BYTE  drive letter (bits 0-6), remote if bit 7 set
 01h 11 BYTEs search template
 0Ch  BYTE  search attributes
---DOS 2.x (and some DOS 3.x???)---
 00h  BYTE  search attributes
 01h  BYTE  drive letter
 02h 11 BYTEs search template
---WILDUNIX.COM---
 00h 12 BYTEs 15-character wildcard search pattern and drive letter (packed)
 0Ch  BYTE  search attributes
---DOS 2.x and most 3.x---
 0Dh  WORD  entry count within directory
 0Fh  DWORD pointer to DTA???
 13h  WORD  cluster number of start of parent directory
---PC-DOS 4.01, MS-DOS 3.2/3.3/5.0---
 0Dh  WORD  entry count within directory
 0Fh  WORD  cluster number of start of parent directory
 11h  4 BYTEs reserved
---OS/2 MVDM---
 00h  WORD  "OS2_BMP_handle"
 02h  WORD  "OS2_LastEnt"
 04h  DWORD "OS2_Checksum"
 08h  BYTE  "OS2_usi_flag"
 09h  DWORD used by DOS emulator for second pass for volume-label searches
 0Dh  WORD  (ret) "DOS_LastEnt" entry count within directory
 0Fh  BYTE  OS/2 Processed-FindFirst flag
    00h FindFirst processed by DOS
    42h FindFirst processed by OS/2
```

```
 10h  5 BYTEs reserved for future use
---all versions, documented fields---
 15h  BYTE  attribute of file found
 16h  WORD  file time (see #01665 at AX=5700h)
 18h  WORD  file date (see #01666 at AX=5700h)
 1Ah  DWORD file size
 1Eh 13 BYTEs ASCIZ filename+extension
--------f-214E-------------------------------
INT 21 - WILDUNIX.COM internal - INSTALLATION CHECK
  AH = 4Eh
  DS:DX = 0000h:0000h
Return: AH = 99h if installed
Program: WILDUNIX.COM is a resident Unix-style wildcard expander by Steve
    Hosgood and Terry Barnaby
--------D-214F-------------------------------
INT 21 - DOS 2+ - "FINDNEXT" - FIND NEXT MATCHING FILE
  AH = 4Fh
  Disk Transfer Area contains data block from previous FindFirst or
    FindNext call
Return: CF clear if successful
      Disk Transfer Area updated
  CF set on error
      AX = error code (12h) (see #01680 at AH=59h/BX=0000h)
Notes:  under Novell DOS 7, if the FindFirst call (AH=4Eh) had CX=0088h, then
    the next matching deleted file will be returned
  since the entire state of a FindFirst/FindNext sequence is contained
    in the data block in the DTA, other disk operations such as renaming,
    moving, deleting, or creating files can cause inaccurate directory
    searches, such as finding the same file twice
BUG:  DR DOS 3.41 and 5.0 return the Directory attribute for the volume label
SeeAlso: AH=12h,AH=4Eh,AX=714Fh,AX=71A1h
--------D-2150-------------------------------
INT 21 - DOS 2+ internal - SET CURRENT PROCESS ID (SET PSP ADDRESS)
  AH = 50h
  BX = segment of PSP for new process
Notes:  DOS uses the current PSP address to determine which processes own files
    and memory; it corresponds to process identifiers used by other OSs
  under DOS 2.x, this function cannot be invoked inside an INT 28h
    handler without setting the Critical Error flag
  under MS-DOS 3.0+ and DR DOS 3.41+, this function does not use any of
    the DOS-internal stacks and may thus be called at any time, even
```

```
    during another INT 21h call
  some Microsoft applications such as Quick C 2.51 use segments of 0000h
    and FFFFh and direct access to the SDA (see #01687 at AX=5D06h) to
    test whether they are running under MS-DOS rather than a compatible
    OS; although one should only call this function with valid PSP
    addresses, any program hooking it should be prepared to handle
    invalid addresses
  this function is supported by the OS/2 compatibility box
  this call was undocumented prior to the release of DOS 5.0
SeeAlso: AH=26h,AH=51h,AH=62h
--------v-2150FD-----------------------------
INT 21 - VIRUS - "Predator 2" - INSTALLATION CHECK
  AX = 50FDh
Return: AX = FD50h if resident
SeeAlso: AX=4BFFh"VIRUS",AX=5454h"VIRUS"
--------D-2151-----------------------------
INT 21 - DOS 2+ internal - GET CURRENT PROCESS ID (GET PSP ADDRESS)
  AH = 51h
Return: BX = segment of PSP for current process
Notes:  DOS uses the current PSP address to determine which processes own files
    and memory; it corresponds to process identifiers used by other OSs
  under DOS 2.x, this function cannot be invoked inside an INT 28h
    handler without setting the Critical Error flag
  under DOS 3.0+, this function does not use any of the DOS-internal
    stacks and may thus be called at any time, even during another
    INT 21h call
  supported by OS/2 compatibility box
  identical to the documented AH=62h
  this call was undocumented prior to the release of DOS 5.0
SeeAlso: AH=26h,AH=50h,AH=62h
--------D-2152-----------------------------
INT 21 U - DOS 2+ internal - "SYSVARS" - GET LIST OF LISTS
  AH = 52h
Return: ES:BX -> DOS list of lists (see #01627)
Notes:  partially supported by OS/2 v1.1 compatibility box (however, most
    pointers are FFFFh:FFFFh, LASTDRIVE is FFh, and the NUL header "next"
    pointer is FFFFh:FFFFh).
  partially supported by the Windows NT DOS box; contains only a
    rudimentary Current Directory Structure (see #01645)
  on return, ES points at the DOS data segment (see also INT 2F/AX=1203h)
  Quarterdeck's suggested check for the use of its DOSDATA.SYS or
```

```
    DOS-UP.SYS is to test whether the list-of-lists segment is greater
    than the segment of the first memory block; a better check for
    DOS-UP.SYS is INT 21/AX=2B01h/CX=444Dh
  because not all DOS workalikes support all fields in the List of Lists,
    applications should ensure that pointers are neither 0000h:0000h
    nor FFFFh:FFFFh before using them
  Windows for Workgroups 3.11 network and Windows95 set the path to
    the local drive and directory even for network drives; in that
    case the UNC form \\SERVER\SHARE can be obtained with
    INT 21/AX=5F02h or INT 21/AX=5F46h. LapLink RemoteAccess does the
    same even for INT 21/AX=5F02h
  Windows95 GUI no longer returns the true path for a SUBSTed drive,
    but MS-DOS 7.00 does; use INT 21/AH=60h to obtain the true name
SeeAlso: INT 2F/AX=1203h

Format of List of Lists:
Offset  Size  Description (Table 01627)
 -24  WORD  (DOS 3.1+) contents of CX from INT 21/AX=5E01h
 -22  WORD  (DOS ???+) LRU counter for FCB caching
 -20  WORD  (DOS ???+) LRU counter for FCB opens
 -18  DWORD (DOS ???+) address of OEM function handler (see INT 21/AH=F8h)
      FFFFh:FFFFh if not installed or not available
 -14  WORD  (DOS ???+) offset in DOS CS of code to return from INT 21 call
 -12  WORD  (DOS 3.1+) sharing retry count (see AX=440Bh)
 -10  WORD  (DOS 3.1+) sharing retry delay (see AX=440Bh)
 -8 DWORD (DOS 3.0+) pointer to current disk buffer
 -4 WORD  (DOS 3.0+) pointer in DOS data segment of unread CON input
    when CON is read via a handle, DOS reads an entire line,
      and returns the requested portion, buffering the rest
      for the next read.  0000h indicates no unread input
 -2 WORD  segment of first memory control block (see #01628)
 00h  DWORD pointer to first Drive Parameter Block (see #01395 at AH=32h)
 04h  DWORD -> first System File Table (see #01639,#01640,#01641,#01642)
 08h  DWORD pointer to active CLOCK$ device's header (most recently loaded
      driver with CLOCK bit set)
 0Ch  DWORD pointer to active CON device's header (most recently loaded
      driver with STDIN bit set)
---DOS 2.x---
 10h  BYTE  number of logical drives in system
 11h  WORD  maximum bytes/block of any block device
 13h  DWORD pointer to first disk buffer (see #01649,#01650)
```

```
 17h 18 BYTEs actual NUL device driver header (not a pointer!)
    NUL is always the first device on DOS's linked list of device
      drivers. (see #01646)
---DOS 3.0---
 10h  BYTE  number of block devices
 11h  WORD  maximum bytes/block of any block device
 13h  DWORD pointer to first disk buffer (see #01650,#01652)
 17h  DWORD pointer to array of current directory structures (see #01643)
 1Bh  BYTE  value of LASTDRIVE command in CONFIG.SYS (default 5)
 1Ch  DWORD pointer to STRING= workspace area
 20h  WORD  size of STRING area (the x in STRING=x from CONFIG.SYS)
 22h  DWORD pointer to FCB table
 26h  WORD  the y in FCBS=x,y from CONFIG.SYS
 28h 18 BYTEs actual NUL device driver header (not a pointer!)
    NUL is always the first device on DOS's linked list of device
      drivers. (see #01646)
---DOS 3.1-3.3---
 10h  WORD  maximum bytes per sector of any block device
 12h  DWORD pointer to first disk buffer in buffer chain (see #01650)
 16h  DWORD pointer to array of current directory structures (see #01643)
 1Ah  DWORD pointer to system FCB tables (see #01640,#01641,#01642)
 1Eh  WORD  number of protected FCBs (the y in the CONFIG.SYS FCBS=x,y)
 20h  BYTE  number of block devices installed
 21h  BYTE  number of available drive letters (largest of 5, installed
      block devices, and CONFIG.SYS LASTDRIVE=).  Also size of
      current directory structure array.
 22h 18 BYTEs actual NUL device driver header (not a pointer!)
    NUL is always the first device on DOS's linked list of device
      drivers. (see #01646)
 34h  BYTE  number of JOIN'ed drives
---DOS 4.x---
 10h  WORD  maximum bytes per sector of any block device
 12h  DWORD pointer to disk buffer info record (see #01652,#01653)
    Note: although the initialization code in IO.SYS uses this
      pointer, MSDOS.SYS does not, instead using the hardcoded
      address of the info record
 16h  DWORD pointer to array of current directory structures
    (see #01643,#01644)
 1Ah  DWORD pointer to system FCB tables (see #01640,#01641,#01642)
 1Eh  WORD  number of protected FCBs (the y in the CONFIG.SYS FCBS=x,y)
    (always 00h for DOS 5.0)
```

```
 20h  BYTE  number of block devices installed
 21h  BYTE  number of available drive letters; also size of current
      directory structure array.
    For DOS 4.0-6.0: largest of 5, installed block devices,
      and CONFIG.SYS LASTDRIVE=
    For DOS 7.x (Windows9X), set to 32 if no LASTDRIVE= or
      LASTDRIVEHIGH=, else set to larger of installed block
      devices and LASTDRIVE(HIGH)=
 22h 18 BYTEs actual NUL device driver header (not a pointer!)
    NUL is always the first device on DOS's linked list of device
      drivers. (see #01646)
 34h  BYTE  number of JOIN'ed drives
 35h  WORD  pointer within IBMDOS code segment to list of special program
      names (see #01662)
    (always 0000h for DOS 5.0)
 37h  DWORD pointer to FAR routine for resident IFS utility functions
    (see #01658)
    may be called by any IFS driver which does not wish to
      service functions 20h or 24h-28h itself
 3Bh  DWORD pointer to chain of IFS (installable file system) drivers
 3Fh  WORD  the x in BUFFERS x,y (rounded up to multiple of 30 if in EMS)
 41h  WORD  number of lookahead buffers (the y in BUFFERS x,y)
 43h  BYTE  boot drive (1=A:)
 44h  BYTE  flag: 01h to use DWORD moves (80386+), 00h otherwise
 45h  WORD  extended memory size in KB
---DOS 5.0-6.0---
 10h 39 BYTEs as for DOS 4.x (above)
 37h  DWORD pointer to SETVER program list or 0000h:0000h
 3Bh  WORD  (DOS=HIGH) offset in DOS CS of function to fix A20 control
      when executing special .COM format
 3Dh  WORD  PSP of most-recently EXECed program if DOS in HMA, 0000h if low
    used for maintaining count of INT 21 calls which disable A20
      on return
 3Fh  8 BYTEs as for DOS 4.x (above)
---Windows NT DOS Box---
 10h  6 BYTEs ???
 16h  DWORD pointer to array of current directory structures (see #01645)
 1Ah  6 BYTEs ???
 20h  BYTE  number of block devices installed
 21h  BYTE  number of local drive letters (= installed block devices)
    Also size of current directory structure array.
```

```
 22h 18 BYTEs actual NUL device driver header (not a pointer!)
    NUL is always the first device on DOS's linked list of device
      drivers. (see #01646)
---DOS 7.x---
 10h 55 BYTEs as for DOS 5.0-6.0 (above)
 47h 25 BYTEs ???
 60h  BYTE  "DOS_FLAG" (see also INT 26)

Format of DOS memory control block:
Offset  Size  Description (Table 01628)
 00h  BYTE  block type: 5Ah if last block in chain, otherwise 4Dh
 01h  WORD  PSP segment of owner or special flag value (see #01629)
 03h  WORD  size of memory block in paragraphs
 05h  3 BYTEs unused by MS-DOS
    (386MAX) if locked-out block, region start/prev region end
---DOS 2.x,3.x---
 08h  8 BYTEs unused
---DOS 4.0+ ---
 08h  8 BYTEs ASCII program name if PSP memory block or DR DOS UMB,
      else garbage
    null-terminated if less than 8 characters
Notes:  the next MCB is at segment (current + size + 1)
  under DOS 3.1+, the first memory block is the DOS data segment,
    containing installable drivers, buffers, etc.  Under DOS 4.0+ it is
    divided into subsegments, each with its own memory control block
    (see #01633), the first of which is at offset 0000h.
  for DOS 5+, blocks owned by DOS may have either "SC" or "SD" in bytes
    08h and 09h.  "SC" is system code or locked-out inter-UMB memory,
    "SD" is system data, device drivers, etc.
  Some versions of DR DOS use only seven characters of the program name,
    placing a NUL in the eighth byte.
SeeAlso: #01630,#01632,#01633

(Table 01629)
Values for special flag PSP segments:
 0000h  free
 0006h  DR DOS XMS UMB
 0007h  DR DOS excluded upper memory ("hole")
 0008h  belongs to DOS
 FFF7h  386MAX v6.01+ ???
 FFFAh  386MAX UMB control block (see #01477 at AX=4402h"386MAX")
```

```
 FFFDh  386MAX locked-out memory
 FFFEh  386MAX UMB (normally immediately follows its control block)
 FFFFh  386MAX v6.01+ device driver

Format of MS-DOS 5+ UMB control block:
Offset  Size  Description (Table 01630)
 00h  BYTE  type: 5Ah if last block in chain, 4Dh otherwise
 01h  WORD  first available paragraph in UMB if control block at start
      of UMB, 000Ah if control block at end of UMB
 03h  WORD  length in paragraphs of following UMB or locked-out region
 05h  3 BYTEs unused
 08h  8 BYTEs block type name: "UMB" if start block, "SM" if end block in UMB
SeeAlso: #01628,#01631

Format of MS-DOS 7.0 HMA memory control block:
Offset  Size  Description (Table 01631)
 00h  WORD  signature "MS" (4Dh 53h)
 02h  WORD  usage flag???
    0000h free
    else ???
 04h  WORD  size of memory block in bytes (not counting MCB)
 06h  WORD  offset of next memory block in HMA or 0000h
 08h  8 BYTEs unused (0)
SeeAlso: #01628,#01630,#01632

Format of STARLITE (General Software's Embedded DOS) memory control block:
Offset  Size  Description (Table 01632)
 00h  BYTE  block type: 5Ah if last block in chain, otherwise 4Dh
 01h  WORD  PSP segment of owner, 0000h if free, 0008h if belongs to DOS
 03h  WORD  size of memory block in paragraphs
 05h  BYTE  unused
 06h  WORD  segment address of next memory control block (0000h if last)
 08h  WORD  segment address of previous memory control block or 0000h
 0Ah  6 BYTEs reserved

Format of DOS 4.0+ data segment subsegment control blocks:
Offset  Size  Description (Table 01633)
 00h  BYTE  subsegment type (blocks typically appear in this order)
    "D"  device driver
    "E"  device driver appendage
    "I"  IFS (Installable File System) driver
```

```
        (MS-DOS 7) high-loaded drive data table array (see #02603)
    "F"  FILES=  control block storage area (for FILES>5)
    "X"  FCBS=   control block storage area, if present
    "C"  BUFFERS EMS workspace area (if BUFFERS /X option used)
    "B"  BUFFERS=  storage area
    "L"  LASTDRIVE=  current directory structure array storage area
    "S"  STACKS=  code and data area, if present (see #01634,#01635)
    "T"  INSTALL= transient code
 01h  WORD  paragraph of subsegment start (usually the next paragraph)
 03h  WORD  size of subsegment in paragraphs
 05h  3 BYTEs unused
 08h  8 BYTEs for types "D" and "I", base name of file from which the driver
      was loaded (unused for other types)

Format of data at start of STACKS code segment (if present):
Offset  Size  Description (Table 01634)
 00h  WORD  ???
 02h  WORD  number of stacks (the x in STACKS=x,y)
 04h  WORD  size of stack control block array (should be 8*x)
 06h  WORD  size of each stack (the y in STACKS=x,y)
 08h  DWORD pointer to STACKS data segment
 0Ch  WORD  offset in STACKS data segment of stack control block array
 0Eh  WORD  offset in STACKS data segment of last element of that array
 10h  WORD  offset in STACKS data segment of the entry in that array for
      the next stack to be allocated (initially same as value in
      0Eh and works its way down in steps of 8 to the value in
      0Ch as hardware interrupts pre-empt each other)
Note: the STACKS code segment data may, if present, be located as follows:
    DOS 3.2:  The code segment data is at a paragraph boundary fairly early
      in the IBMBIO segment (seen at 0070:0190h)
    DOS 3.3:  The code segment is at a paragraph boundary in the DOS data
      segment, which may be determined by inspecting the segment
      pointers of the vectors for those of interrupts 02h, 08h-0Eh,
      70h, 72-77h which have not been redirected by device drivers
      or TSRs.
    DOS 4.0+  Identified by sub-segment control block type "S" within the DOS
      data segment.
SeeAlso: #01636,INT B4"STACKMAN"

Format of array elements in STACKS data segment:
Offset  Size  Description (Table 01635)
```

```
 00h  BYTE  status: 00h=free, 01h=in use, 03h=corrupted by overflow of
      higher stack.
 01h  BYTE  not used
 02h  WORD  previous SP
 04h  WORD  previous SS
 06h  WORD  ptr to word at top of stack (new value for SP). The word at the
      top of the stack is preset to point back to this control
      block.

Format of SHARE.EXE hooks (DOS 3.1-6.00):
Offset  Size  Description (Table 01636)
(offsets from first system file table--pointed at by ListOfLists+04h)
-3Ch  DWORD pointer to FAR routine for ???
    Note: not called by MS-DOS 3.3, set to 0000h:0000h by
      SHARE 3.3+
-38h  DWORD pointer to FAR routine called on opening file
    on call, internal DOS location points at filename
      (see #01687 at AX=5D06h)
    Return: CF clear if successful
      CF set on error
          AX = DOS error code (24h)
          (see #01680 at AH=59h/BX=0000h)
    Note: SHARE directly accesses DOS-internal data to get name of
      file just opened
-34h  DWORD pointer to FAR routine called on closing file
    ES:DI -> system file table
    Note: does something to every Record Lock Record for file
-30h  DWORD pointer to FAR routine to close all files for given computer
    (called by AX=5D03h)
-2Ch  DWORD pointer to FAR routine to close all files for given process
    (called by AX=5D04h)
-28h  DWORD pointer to FAR routine to close file by name
    (called by AX=5D02h)
    DS:SI -> DOS parameter list (see #01686 at AX=5D00h)
       DPL's DS:DX -> name of file to close
    Return: CF clear if successful
      CF set on error
          AX = DOS error code (03h)
          (see #01680 at AH=59h/BX=0000h)
-24h  DWORD pointer to FAR routine to lock region of file
    call with BX = file handle
```

```
        ---DOS 3.x---
        CX:DX = starting offset
        SI:AX = size
        ---DOS 4.0+ ---
        DS:DX -> lock range
          DWORD start offset
          DWORD size in bytes
    Return: CF set on error
          AL = DOS error code (21h) (see #01680 at AH=59h)
    Note: not called if file is marked as remote
-20h  DWORD pointer to FAR routine to unlock region of file
    call with BX = file handle
        ---DOS 3.x---
        CX:DX = starting offset
        SI:AX = size
        ---DOS 4.0+ ---
        DS:DX -> lock range
          DWORD start offset
          DWORD size in bytes
    Return: CF set on error
          AL = DOS error code (21h) (see #01680 at AH=59h)
    Note: not called if file is marked as remote
-1Ch  DWORD pointer to FAR routine to check if file region is locked
    call with ES:DI -> system file table entry for file
      CX = length of region from current position in file
    Return: CF set if any portion of region locked
          AX = 0021h
-18h  DWORD pointer to FAR routine to get open file list entry
    (called by AX=5D05h)
    call with DS:SI -> DOS parameter list (see #01686 at AX=5D00h)
      DPL's BX = index of sharing record
      DPL's CX = index of SFT in SFT chain of sharing rec
    Return: CF set on error or not loaded
          AX = DOS error code (12h) (see #01680 at AH=59h)
      CF clear if successful
          ES:DI -> filename
          CX = number of locks owned by specified SFT
          BX = network machine number
          DX destroyed
-14h  DWORD pointer to FAR routine for updating FCB from SFT???
    call with DS:SI -> unopened FCB
```

```
        ES:DI -> system file table entry
    Return: BL = C0h???
    Note: copies following fields from SFT to FCB:
       starting cluster of file   0Bh  1Ah
       sharing record offset    33h  1Ch
       file attribute     04h  1Eh
-10h  DWORD pointer to FAR routine to get first cluster of FCB file ???
    call with ES:DI -> system file table entry
        DS:SI -> FCB
    Return: CF set if SFT closed or sharing record offsets
        mismatched
      CF clear if successful
          BX = starting cluster number from FCB
-0Ch  DWORD pointer to FAR routine to close file if duplicate for process
    DS:SI -> system file table
    Return: AX = number of handle in JFT which already uses SFT
    Note: called during open/create of a file
    Note: if SFT was opened with inheritance enabled and sharing
      mode 111, does something to all other SFTs owned by
      same process which have the same file open mode and
      sharing record
-08h  DWORD pointer to FAR routine for closing file
    Note: closes various handles referring to file most-recently
      opened
-04h  DWORD pointer to FAR routine to update directory info in related SFT
      entries
    call with ES:DI -> system file table entry for file (see #01641)
        AX = subfunction (apply to each related SFT)
        00h: update time stamp (offset 0Dh) and date
             stamp (offset 0Fh)
        01h: update file size (offset 11h) and starting
             cluster (offset 0Bh).  Sets last-accessed
             cluster fields to start of file if file
             never accessed
        02h: as function 01h, but last-accessed fields
             always changed
        03h: do both functions 00h and 02h
    Note: follows ptr at offset 2Bh in system file table entries
    Note: NOP if opened with no-inherit or via FCB
Notes:  most of the above hooks (except -04h, -14h, -18h, and -3Ch) assume
    either that SS=DOS DS or SS=DS=DOS DS and directly access
```

```
    DOS-internal data
  sharing hooks are not supported by DR DOS 5-6; they appear to be
    supported by Novell DOS 7, with a segment of 0000h indicating the
    DOS data segment
SeeAlso: #01637,#01638

Format of sharing record:
Offset  Size  Description (Table 01637)
 00h  BYTE  flag
    00h free block
    01h allocated block
    FFh end marker
 01h  WORD  size of block
 03h  BYTE  checksum of pathname (including NUL)
    if sum of ASCII values is N, checksum is (N/256 + N%256)
 04h  WORD  offset in SHARE's DS of first Record Lock Record (see #01638)
 06h  DWORD pointer to start of system file table chain for file
 0Ah  WORD  unique sequence number
 0Ch  var ASCIZ full pathname
Note: not supported by DR DOS SHARE 1.1 and 2.0; will reportedly be
    supported by Novell DOS 7
SeeAlso: #01636,#01638

Format of SHARE.EXE Record Lock Record:
Offset  Size  Description (Table 01638)
 00h  WORD  offset in SHARE's DS of next lock table in list or 0000h
 02h  DWORD offset in file of start of locked region
 06h  DWORD offset in file of end of locked region
 0Ah  DWORD pointer to System File Table entry for this file
 0Eh  WORD  PSP segment of lock's owner
---DOS 5+ ---
 10h  WORD  lock type: (00h lock all, 01h lock writes only)
SeeAlso: #01636,#01637,#01639,#01642

Format of DOS 2.x system file tables:
Offset  Size  Description (Table 01639)
 00h  DWORD pointer to next file table (offset FFFFh if last)
 04h  WORD  number of files in this table
 06h  28h bytes per file
  Offset  Size  Description
   00h  BYTE  number of file handles referring to this file
```

```
   01h  BYTE  file open mode (see #01402 at AH=3Dh)
   02h  BYTE  file attribute
   03h  BYTE  drive (0 = character device, 1 = A, 2 = B, etc)
   04h 11 BYTEs filename in FCB format (no path,no period,blank-padded)
   0Fh  WORD  ???
   11h  WORD  ???
   13h  DWORD file size???
   17h  WORD  file date in packed format (see #01666 at AX=5700h)
   19h  WORD  file time in packed format (see #01665 at AX=5700h)
   1Bh  BYTE  device attribute (see #01423 at AX=4400h)
  ---character device---
   1Ch  DWORD pointer to device driver
  ---block device---
   1Ch  WORD  starting cluster of file
   1Eh  WORD  relative cluster in file of last cluster accessed
  ------
   20h  WORD  absolute cluster number of current cluster
   22h  WORD  ???
   24h  DWORD current file position???
SeeAlso: #01640,#01641,#01642

Format of DOS 3.0 system file tables and FCB tables:
Offset  Size  Description (Table 01640)
 00h  DWORD pointer to next file table (offset FFFFh if last)
 04h  WORD  number of files in this table
 06h  38h bytes per file
  Offset  Size  Description
   00h-1Eh as for DOS 3.1+ (see #01641)
   1Fh  WORD  byte offset of directory entry within sector
   21h 11 BYTEs filename in FCB format (no path/period, blank-padded)
   2Ch  DWORD (SHARE.EXE) pointer to previous SFT sharing same file
   30h  WORD  (SHARE.EXE) network machine number which opened file
      (Windows Enhanced mode DOSMGR uses the virtual machine
        ID as the machine number; see INT 2F/AX=1683h)
   32h  WORD  PSP segment of file's owner (first three entries for
        AUX/CON/PRN contain segment of IO.SYS startup code)
   34h  WORD  (SHARE.EXE) offset in SHARE code seg of share record
   36h  WORD  ??? apparently always 0000h
SeeAlso: #01639,#01641,#01642

Format of DOS 3.1-3.3x, DR DOS 5.0-6.0 system file tables and FCB tables:
```

```
Offset  Size  Description (Table 01641)
 00h  DWORD pointer to next file table (offset FFFFh if last)
 04h  WORD  number of files in this table
 06h  35h bytes per file
  Offset  Size  Description
   00h  WORD  number of file handles referring to this file
   02h  WORD  file open mode (see AX=6C00h, #01402 at AH=3Dh)
      bit 15 set if this file opened via FCB
   04h  BYTE  file attribute (see #01420 at AX=4301h)
   05h  WORD  device info word (see #01423 at AX=4400h)
      bit 15 set if remote file
      bit 14 set means do not set file date/time on closing
      bit 12 set means don't inherit on EXEC
      bits 5-0 drive number for disk files
   07h  DWORD pointer to device driver header if character device
      else pointer to DOS Drive Parameter Block
        (see #01395 at AH=32h)
   0Bh  WORD  starting cluster of file
   0Dh  WORD  file time in packed format (see #01665 at AX=5700h)
      not used for character devices in DR DOS
   0Fh  WORD  file date in packed format (see #01666 at AX=5700h)
      not used for character devices in DR DOS
   11h  DWORD file size
  ---system file table---
   15h  DWORD current offset in file (may be larger than size of
        file; INT 21/AH=42h does not check new position)
  ---FCB table---
   15h  WORD  counter for last I/O to FCB
   17h  WORD  counter for last open of FCB
      (these are separate to determine the times of the
        latest I/O and open)
  ---
   19h  WORD  relative cluster within file of last cluster accessed
   1Bh  WORD  absolute cluster number of last cluster accessed
      0000h if file never read or written???
   1Dh  WORD  number of sector containing directory entry
      (see #01352)
   1Fh  BYTE  number of dir entry within sector (byte offset/32)
   20h 11 BYTEs filename in FCB format (no path/period, blank-padded)
   2Bh  DWORD (SHARE.EXE) pointer to previous SFT sharing same file
   2Fh  WORD  (SHARE.EXE) network machine number which opened file
```

```
      (Windows Enhanced mode DOSMGR uses the virtual machine
        ID as the machine number; see INT 2F/AX=1683h)
   31h  WORD  PSP segment of file's owner (see #01378 at AH=26h)
        (first three entries for AUX/CON/PRN contain segment
        of IO.SYS startup code)
   33h  WORD  offset within SHARE.EXE code segment of
        sharing record (see #01637) 0000h = none
SeeAlso: #01639,#01640,#01642

Format of DOS 4.0-6.0 system file tables and FCB tables:
Offset  Size  Description (Table 01642)
 00h  DWORD pointer to next file table (offset FFFFh if last)
 04h  WORD  number of files in this table
 06h  3Bh bytes per file
  Offset  Size  Description
   00h  WORD  number of file handles referring to this file
      FFFFh if in use but not referenced
   02h  WORD  file open mode (see AX=6C00h,#01402 at AH=3Dh)
      bit 15 set if this file opened via FCB
   04h  BYTE  file attribute (see #01420 at AX=4301h)
   05h  WORD  device info word (see also #01423 at AX=4400h)
      bit 15 set if remote file
      bit 14 set means do not set file date/time on closing
      bit 13 set if named pipe
      bit 12 set if no inherit
      bit 11 set if network spooler
      bit 7  set if device, clear if file (only if local)
      bits 6-0 as for AX=4400h
   07h  DWORD pointer to device driver header if character device
      else pointer to DOS Drive Parameter Block
        (see #01395 at AH=32h) or REDIR data
   0Bh  WORD  starting cluster of file (local files only)
      not set for FAT32-capable versions of Windows95
        (since they allow 32-bit cluster numbers)
   0Dh  WORD  file time in packed format (see #01665)
   0Fh  WORD  file date in packed format (see #01666)
   11h  DWORD file size
   15h  DWORD current offset in file (SFT)
      LRU counters (FCB table, two WORDs)
  ---local file---
   19h  WORD  relative cluster within file of last cluster accessed
```

```
   1Bh  DWORD number of sector containing directory entry
   1Fh  BYTE  number of dir entry within sector (byte offset/32)
  ---network redirector---
   19h  DWORD pointer to REDIRIFS record
   1Dh  3 BYTEs ???
  ------
   20h 11 BYTEs filename in FCB format (no path/period, blank-padded)
   2Bh  DWORD (SHARE.EXE) pointer to previous SFT sharing same file
   2Fh  WORD  (SHARE.EXE) network machine number which opened file
      (Windows Enhanced mode DOSMGR uses the virtual machine
        ID as the machine number; see INT 2F/AX=1683h)
   31h  WORD  PSP segment of file's owner (see #01378 at AH=26h)
        (first three entries for AUX/CON/PRN contain segment
        of IO.SYS startup code)
   33h  WORD  offset within SHARE.EXE code segment of
      sharing record (see #01595)  0000h = none
   35h  WORD  (local) absolute cluster number of last clustr accessed
      (redirector) ???
   37h  DWORD pointer to IFS driver for file, 0000000h if native DOS
Note: the OS/2 2.0 DOS Boot Session does not properly fill in the filename
    field due to incomplete support for SFTs; the OS/2 2.0 DOS Window
    does not appear to support SFTs at all
SeeAlso: #01639,#01640,#01641

Format of current directory structure (CDS) (array, LASTDRIVE entries):
Offset  Size  Description (Table 01643)
 00h 67 BYTEs ASCIZ path in form X:\PATH (local) or UNC form
      \\SERVER\PATH (network, see notes below)
 43h  WORD  drive attributes (also see note below) (see also AX=5F07h)
    bit 15: uses network redirector  \ invalid if 00, installable
    bit 14: physical drive     / file system if 11
    bit 13: JOIN'ed   \ path above is true path that would be
    bit 12: SUBST'ed  / needed if not under SUBST or JOIN
    bit 11: removable media (WinNT4.0)
    bit 10: hard disk??? (WinNT4.0)
    bit  7: remote drive hidden from redirector's assign-list and
        exempt from network connection make/break commands;
        set for CD-ROM drives by MSCDEX (not supported by
        CORELCDX)
 45h  DWORD pointer to Drive Parameter Block for drive
      (see #01395 at AH=32h)
```

```
---local drives---
 49h  WORD  starting cluster of current directory
    0000h = root, FFFFh = never accessed
 4Bh  WORD  ??? seems to be FFFFh always
 4Dh  WORD  ??? seems to be FFFFh always
---network drives---
 49h  DWORD pointer to redirector or REDIRIFS record, or FFFFh:FFFFh
    (DOS 4 only) available for use by IFS driver
 4Dh  WORD  stored user data from INT 21/AX=5F03h
------
 4Fh  WORD  offset in current directory path of backslash corresponding to
      root directory for drive
    this value specifies how many characters to hide from the
      "CHDIR" and "GETDIR" calls; normally set to 2 to hide the
      drive letter and colon, SUBST, JOIN, and networks change it
      so that only the appropriate portion of the true path is
      visible to the user
---DOS 4.0+ ---
 51h  BYTE  (DOS 4 only, remote drives) device type
    04h network drive
 52h  DWORD pointer to IFS driver (DOS 4) or redirector block (DOS 5+) for
      this drive, 00000000h if native DOS
 56h  WORD  available for use by IFS driver
Notes:  the path for invalid drives is normally set to X:\, but may be empty
    after JOIN x: /D in DR DOS 5.0 or NET USE x: /D in older LAN versions
  normally, only one of bits 13&12 may be set together with bit 14, but
    DR DOS 5.0 uses other combinations for bits 15-12: 0111 JOIN,
    0001 SUBST, 0101 ASSIGN (see #01644)
  Windows for Workgroups 3.11 network sets the path to the local
    drive and directory even for network drives; in that case the
    UNC form \\SERVER\SHARE can be obtained with INT 21/AX=5F02h
    or INT 21/AX=5F46h. LapLink RemoteAccess does the same even for
    INT 21/AX=5F02h
  Window NT 4.0 always uses X:\PATH, even for substituted drives (i.e.
    the CDS will *not* contain the original path prior to SUBST)
  SoftWindows on the Macintosh PowerPC sets \\E for the host drive
SeeAlso: #01644,INT 21/AX=71AAh/BL=00h

Format of DR DOS 5.0-6.0 current directory structure entry (array):
Offset  Size  Description (Table 01644)
 00h 67 BYTEs ASCIZ pathname of actual root directory for this logical drive
```

```
 43h  WORD  drive attributes
    1000h SUBSTed drive
    3000h??? JOINed drive
    4000h physical drive
    5000h ASSIGNed drive
    7000h JOINed drive
    8000h network drive
 45h  BYTE  physical drive number (0=A:) if this logical drive is valid
 46h  BYTE  ??? apparently flags for JOIN and ASSIGN
 47h  WORD  cluster number of start of parent directory (0000h = root)
 49h  WORD  entry number of current directory in parent directory
 4Bh  WORD  cluster number of start of current directory
 4Dh  WORD  used for media change detection (details not available)
 4Fh  WORD  cluster number of SUBST/JOIN "root" directory
    0000h if physical root directory
SeeAlso: #01643

Format of Windows NT Current Directory Structure (CDS) (array):
Offset  Size  Description (Table 01645)
 00h 67 BYTEs ASCIZ path in form X:\ (does not show either current directory
      or network path)
 43h  4 BYTEs ???
Note: the WinNT CDS contains only as many entries as there are local drives,
    not LASTDRIVE entries.

Format of DOS device driver header:
Offset  Size  Description (Table 01646)
 00h  DWORD pointer to next driver, offset=FFFFh if last driver
 04h  WORD  device attributes (see #01647,#01648)
 06h  WORD  device strategy entry point
    call with ES:BX -> request header
      (see #02597 at INT 2F/AX=0802h)
 08h  WORD  device interrupt entry point
---character device---
 0Ah  8 BYTEs blank-padded character device name
---block device---
 0Ah  BYTE  number of subunits (drives) supported by driver
 0Bh  7 BYTEs normally unused; sometimes contains signature to indicate
      specific drivers:
    "$PCMATA"   PCMCIA driver PCMATA.SYS (see AX=440Dh"DOS 3.2+")
    "AHADDVR"   Adaptec SCSI disk driver ASPIDISK.SYS
```

```
    "DBLSPAC"   MS DoubleSpace or DriveSpace
    "DSKREET"   NortonUtils v5+ Diskreet (see INT 2F/AX=FE00h)
    "GFS  "   LapLink III device driver DD.BIN
    "SIDExxx"   PCMCIA driver ATADRV.EXE (see AX=440Dh"DOS 3.2+")
    "STAC-CD"   Stacker/Stacker Anywhere (see AX=4404h"Stacker")
---DoubleSpace/DriveSpace---
 12h  2 BYTEs signature ",." (2Ch 2Eh)
 14h  var preloading API entry point
---CD-ROM---
 12h  WORD  reserved, must be 0000h
    appears to be another device chain
 14h  BYTE  drive letter, 01h=A:, etc. (must initially be 00h; this byte
      is set by MSCDEX when it loads)
 15h  BYTE  number of units
 16h  6 BYTEs signature 'MSCDnn' where 'nn' is version (currently '00')
    (this field does not exist in most drivers)
SeeAlso: #02550 at INT 25/AX=CDCDh,#02845 at INT 2F/AX=5600h

Bitfields for device attributes (character device):
Bit(s)  Description (Table 01647)
 15 set (indicates character device)
 14 IOCTL supported (see AH=44h)
 13 (DOS 3.0+) output until busy supported
 12 reserved
 11 (DOS 3.0+) OPEN/CLOSE/RemMedia calls supported
 10-8 reserved
 7  (DOS 5.0+) Generic IOCTL check call supported (driver command 19h)
  (see AX=4410h,AX=4411h)
 6  (DOS 3.2+) Generic IOCTL call supported (driver command 13h)
  (see AX=440Ch,AX=440Dh"DOS 3.2+")
 5  reserved
 4  device is special (use INT 29 "fast console output")
 3  device is CLOCK$ (all reads/writes use transfer record described
    below)
 2  device is NUL
 1  device is standard output
 0  device is standard input
Note: for European MS-DOS 4.0, bit 11 also indicates that bits 8-6 contain a
    version code (000 = DOS 3.0,3.1; 001 = DOS 3.2;
    010 = European DOS 4.0)
SeeAlso: #01648,#01646
```

```
Bitfields for device attributes (block device):
Bit(s)  Description (Table 01648)
 15 clear (indicates block device)
 14 IOCTL supported
 13 non-IBM format
 12 network device (device is remote)
 11 (DOS 3.0+) OPEN/CLOSE/RemMedia calls supported
 10 reserved
 9  direct I/O not allowed??? (set by DOS 3.3 DRIVER.SYS for "new" drives)
 8  ??? set by DOS 3.3 DRIVER.SYS for "new" drives
 7  (DOS 5.0+) Generic IOCTL check call supported (driver command 19h)
  (see AX=4410h,AX=4411h)
 6  (DOS 3.2+) Generic IOCTL call supported (driver command 13h)
  implies support for commands 17h and 18h
    (see AX=440Ch,AX=440Dh"DOS 3.2+",AX=440Eh,AX=440Fh)
 5-2  reserved
 1  driver supports 32-bit sector addressing (DOS 3.31+)
 0   reserved
Note: for European MS-DOS 4.0, bit 11 also indicates that bits 8-6 contain a
    version code (000 = DOS 3.0,3.1; 001 = DOS 3.2;
    010 = European DOS 4.0)
SeeAlso: #01647,#01646

Format of DOS 2.x disk buffer:
Offset  Size  Description (Table 01649)
 00h  DWORD pointer to next disk buffer, offset = FFFFh if last
    least-recently used buffer is first in chain
 04h  BYTE  drive (0=A, 1=B, etc), FFh if not in use
 05h  3 BYTEs unused??? (seems always to be 00h 00h 01h)
 08h  WORD  logical sector number
 0Ah  BYTE  number of copies to write (1 for non-FAT sectors)
 0Bh  BYTE  sector offset between copies if multiple copies to be written
 0Ch  DWORD pointer to DOS Drive Parameter Block (see #01395 at AH=32h)
 10h    buffered data
SeeAlso: #01650,#01652,#01653,#01655

Format of DOS 3.x disk buffer:
Offset  Size  Description (Table 01650)
 00h  DWORD pointer to next disk buffer, offset = FFFFh if last
    least-recently used buffer is first in chain
```

```
 04h  BYTE  drive (0=A,1=B, etc), FFh if not in use
 05h  BYTE  buffer flags (see #01651)
 06h  WORD  logical sector number
 08h  BYTE  number of copies to write (1 for non-FAT sectors)
 09h  BYTE  sector offset between copies if multiple copies to be written
 0Ah  DWORD pointer to DOS Drive Parameter Block (see #01395 at AH=32h)
 0Eh  WORD  unused??? (almost always 0)
 10h    buffered data
SeeAlso: #01649,#01652,#01653,#01655

Bitfields for DOS 3.x disk buffer flags:
Bit(s)  Description (Table 01651)
 7  ???
 6  buffer dirty
 5  buffer has been referenced
 4  ???
 3  sector in data area
 2  sector in a directory, either root or subdirectory
 1  sector in FAT
 0  boot sector??? (guess)
SeeAlso: #01656

Format of DOS 4.00 (pre UR 25066) disk buffer info:
Offset  Size  Description (Table 01652)
 00h  DWORD pointer to array of disk buffer hash chain heads (see #01654)
 04h  WORD  number of disk buffer hash chains (referred to as NDBCH below)
 06h  DWORD pointer to lookahead buffer, zero if not present
 0Ah  WORD  number of lookahead sectors, else zero (the y in BUFFERS=x,y)
 0Ch  BYTE  00h if buffers in EMS (/X), FFh if not
 0Dh  WORD  EMS handle for buffers, zero if not in EMS
 0Fh  WORD  EMS physical page number used for buffers (usually 255)
 11h  WORD  ??? seems always to be 0001h
 13h  WORD  segment of EMS physical page frame
 15h  WORD  ??? seems always to be zero
 17h  4 WORDs EMS partial page mapping information???
SeeAlso: #01649,#01650,#01653,#01657

Format of DOS 4.01 (from UR 25066 Corrctive Services Disk on) disk buffer info:
Offset  Size  Description (Table 01653)
 00h  DWORD pointer to array of disk buffer hash chain heads (see #01654)
 04h  WORD  number of disk buffer hash chains (referred to as NDBCH below)
```

```
 06h  DWORD pointer to lookahead buffer, zero if not present
 0Ah  WORD  number of lookahead sectors, else zero (the y in BUFFERS=x,y)
 0Ch  BYTE  01h, possibly to distinguish from pre-UR 25066 format
 0Dh  WORD  ??? EMS segment for BUFFERS (only with /XD)
 0Fh  WORD  ??? EMS physical page number of EMS seg above (only with /XD)
 11h  WORD  ??? EMS segment for ??? (only with /XD)
 13h  WORD  ??? EMS physical page number of above (only with /XD)
 15h  BYTE  ??? number of EMS page frames present (only with /XD)
 16h  WORD  segment of one-sector workspace buffer allocated in main memory
      if BUFFERS/XS or /XD options in effect, possibly to avoid DMA
      into EMS
 18h  WORD  EMS handle for buffers, zero if not in EMS
 1Ah  WORD  EMS physical page number used for buffers (usually 255)
 1Ch  WORD  ??? appears always to be 0001h
 1Eh  WORD  segment of EMS physical page frame
 20h  WORD  ??? appears always to be zero
 22h  BYTE  00h if /XS, 01h if /XD, FFh if BUFFERS not in EMS
SeeAlso: #01649,#01650,#01652,#01657

Format of DOS 4.x disk buffer hash chain head (array, one entry per chain):
Offset  Size  Description (Table 01654)
 00h  WORD  EMS logical page number in which chain is resident, -1 if not
      in EMS
 02h  DWORD pointer to least recently used buffer header.  All buffers on
      this chain are in the same segment.
 06h  BYTE  number of dirty buffers on this chain
 07h  BYTE  reserved (00h)
Notes:  buffered disk sectors are assigned to chain N where N is the sector's
    address modulo NDBCH,  0 <= N <= NDBCH-1
  each chain resides completely within one EMS page
  this structure is in main memory even if buffers are in EMS

Format of DOS 4.0-6.0 disk buffer:
Offset  Size  Description (Table 01655)
 00h  WORD  forward ptr, offset only, to next least recently used buffer
 02h  WORD  backward pointer, offset only
 04h  BYTE  drive (0=A,1=B, etc) if bit 7 clear
    SFT index if bit 7 set
    FFh if not in use
 05h  BYTE  buffer flags (see #01656)
 06h  DWORD logical sector number (local buffers only)
```

```
 0Ah  BYTE  number of copies to write
    for FAT sectors, same as number of FATs
    for data and directory sectors, usually 1
 0Bh  WORD  offset in sectors between copies to write for FAT sectors
 0Dh  DWORD pointer to DOS Drive Parameter Block (see #01395 at AH=32h)
 11h  WORD  size of data in buffer if remote buffer (see also #01656)
 13h  BYTE  reserved (padding)
 14h    buffered data
Note: for DOS 4.x, all buffered sectors which have the same hash value
    (computed as the sum of high and low words of the logical sector
    number divided by the number of disk buffer chains) are on the same
    doubly-linked circular chain; for DOS 5+, only a single circular
    chain exists.
  the links consist of offset addresses only, the segment being the same
    for all buffers in the chain.
SeeAlso: #01649,#01650,#01652

Bitfields for DOS 4.0-6.0 disk buffer flags:
Bit(s)  Description (Table 01656)
 7  remote buffer
 6  buffer dirty
 5  buffer has been referenced (reserved in DOS 5+)
 4  search data buffer (only valid if remote buffer)
 3  sector in data area
 2  sector in a directory, either root or subdirectory
 1  sector in FAT
 0  reserved
SeeAlso: #01651

Format of DOS 5.0-6.0 disk buffer info:
Offset  Size  Description (Table 01657)
 00h  DWORD pointer to least-recently-used buffer header (may be in HMA)
    (see #01655)
 04h  WORD  number of dirty disk buffers
 06h  DWORD pointer to lookahead buffer, zero if not present
 0Ah  WORD  number of lookahead sectors, else zero (the y in BUFFERS=x,y)
 0Ch  BYTE  buffer location
    00h base memory, no workspace buffer
    01h HMA, workspace buffer in base memory
 0Dh  DWORD pointer to one-segment workspace buffer in base memory
 11h  3 BYTEs unused
```

```
 14h  WORD  ???
 16h  BYTE  flag: INT 24 fail while making an I/O status call
 17h  BYTE  temp storage for user memory allocation strategy during EXEC
 18h  BYTE  counter: number of INT 21 calls for which A20 is off
 19h  BYTE  bit flags
    bit 0: ???
    bit 1: SWITCHES=/W specified in CONFIG.SYS (don't load
      WINA20.SYS when MS Windows 3.0 starts)
    bit 2: in EXEC state (INT 21/AX=4B05h)
 1Ah  WORD  offset of unpack code start (used only during INT 21/AX=4B05h)
 1Ch  BYTE  bit 0 set iff UMB MCB chain linked to normal MCB chain
 1Dh  WORD  minimum paragraphs of memory required by program being EXECed
 1Fh  WORD  segment of first MCB in upper memory blocks or FFFFh if DOS
      memory chain in base 640K only (first UMB MCB usually at
      9FFFh, locking out video memory with a DOS-owned memory
      block)
    the MCB this word points at contains a valid link into high
      memory even if it is marked with a 'Z' indicating the last
      memory block
 21h  WORD  paragraph from which to start scanning during memory allocation
SeeAlso: #01652,#01653

(Table 01658)
Call IFS utility function entry point with:
  AH = 20h miscellaneous functions
      AL = 00h get date
    Return: CX = year
      DH = month
      DL = day
      AL = 01h get process ID and computer ID
    Return: BX = current PSP segment
      DX = active network machine number
      AL = 05h get file system info
    ES:DI -> 16-byte info buffer
    Return: buffer filled
      Offset  Size  Description
       00h  2 BYTEs unused
       02h  WORD  number of SFTs (actually counts only
          the first two file table arrays)
       04h  WORD  number of FCB table entries
       06h  WORD  number of proctected FCBs
```

```
       08h  6 BYTEs unused
       0Eh  WORD  largest sector size supported
      AL = 06h get machine name
    ES:DI -> 18-byte buffer for name
    Return: buffer filled with name starting at offset 02h
      AL = 08h get sharing retry count
    Return: BX = sharing retry count
      AL = other
    Return: CF set
  AH = 21h get redirection state
    BH = type (03h disk, 04h printer)
    Return: BH = state (00h off, 01h on)
  AH = 22h ??? some sort of time calculation
    AL = 00h ???
        nonzero ???
  AH = 23h ??? some sort of time calculation
  AH = 24h compare filenames
      DS:SI -> first ASCIZ filename
      ES:DI -> second ASCIZ filename
      Return: ZF set if files are same ignoring case and / vs \
  AH = 25h normalize filename
      DS:SI -> ASCIZ filename
      ES:DI -> buffer for result
      Return: filename uppercased, forward slashes changed to backslashes
  AH = 26h get DOS stack
      Return: DS:SI -> top of stack
        CX = size of stack in bytes
  AH = 27h increment InDOS flag
  AH = 28h decrement InDOS flag
Note: IFS drivers which do not wish to implement functions 20h or 24h-28h may
    pass them on to the default handler pointed at by [LoL+37h]
SeeAlso: #01659,#01660

Format of IFS driver list:
Offset  Size  Description (Table 01659)
 00h  DWORD pointer to next driver header
 04h  8 BYTEs IFS driver name (blank padded), as used by FILESYS command
 0Ch  4 BYTEs ???
 10h  DWORD pointer to IFS utility function entry point (see #01658)
    call with ES:BX -> IFS request (see #01660)
 14h  WORD  offset in header's segment of driver entry point
```

```
  ???
SeeAlso: #01658,#01660

Format of IFS request block:
Offset  Size  Description (Table 01660)
 00h  WORD  total size in bytes of request
 02h  BYTE  class of request
    02h ???
    03h redirection
    04h ???
    05h file access
    06h convert error code to string
    07h ???
 03h  WORD  returned DOS error code
 05h  BYTE  IFS driver exit status
    00h success
    01h ???
    02h ???
    03h ???
    04h ???
    FFh internal failure
 06h 16 BYTEs ???
---request class 02h---
 16h  BYTE  function code
    04h ???
 17h  BYTE  unused???
 18h  DWORD pointer to ???
 1Ch  DWORD pointer to ???
 20h  2 BYTEs ???
---request class 03h---
 16h  BYTE  function code
 17h  BYTE  ???
 18h  DWORD pointer to ???
 1Ch  DWORD pointer to ???
 22h  WORD  returned ???
 24h  WORD  returned ???
 26h  WORD  returned ???
 28h  BYTE  returned ???
 29h  BYTE  unused???
---request class 04h---
 16h  DWORD pointer to ???
```

```
 1Ah  DWORD pointer to ???
---request class 05h---
 16h  BYTE  function code
    01h flush disk buffers
    02h get disk space
    03h MKDIR
    04h RMDIR
    05h CHDIR
    06h delete file
    07h rename file
    08h search directory
    09h file open/create
    0Ah LSEEK
    0Bh read from file
    0Ch write to file
    0Dh lock region of file
    0Eh commit/close file
    0Fh get/set file attributes
    10h printer control
    11h ???
    12h process termination
    13h ???
  ---class 05h function 01h---
   17h  7 BYTEs ???
   1Eh  DWORD pointer to ???
   22h  4 BYTEs ???
   26h  BYTE  ???
   27h  BYTE  ???
  ---class 05h function 02h---
   17h  7 BYTEs ???
   1Eh  DWORD pointer to ???
   22h  4 BYTEs ???
   26h  WORD  returned total clusters
   28h  WORD  returned sectors per cluster
   2Ah  WORD  returned bytes per sector
   2Ch  WORD  returned available clusters
   2Eh  BYTE  returned ???
   2Fh  BYTE  ???
  ---class 05h functions 03h,04h,05h---
   17h  7 BYTEs ???
   1Eh  DWORD pointer to ???
```

```
 22h  4 BYTEs ???
 26h  DWORD pointer to directory name
---class 05h function 06h---
 17h  7 BYTEs ???
 1Eh  DWORD pointer to ???
 22h  4 BYTEs ???
 26h  WORD  attribute mask
 28h  DWORD pointer to filename
---class 05h function 07h---
 17h  7 BYTEs ???
 1Eh  DWORD pointer to ???
 22h  4 BYTEs ???
 26h  WORD  attribute mask
 28h  DWORD pointer to source filespec
 2Ch  DWORD pointer to destination filespec
---class 05h function 08h---
 17h  7 BYTEs ???
 1Eh  DWORD pointer to ???
 22h  4 BYTEs ???
 26h  BYTE  00h FINDFIRST
    01h FINDNEXT
 28h  DWORD pointer to FindFirst search data + 01h if FINDNEXT
 2Ch  WORD  search attribute if FINDFIRST
 2Eh  DWORD pointer to filespec if FINDFIRST
---class 05h function 09h---
 17h  7 BYTEs ???
 1Eh  DWORD pointer to ???
 22h  DWORD pointer to IFS open file structure (see #01661)
 26h  WORD  ???  \ together, specify open vs. create, whether or
 28h  WORD  ???  / not to truncate
 2Ah  4 BYTEs ???
 2Eh  DWORD pointer to filename
 32h  4 BYTEs ???
 36h  WORD  file attributes on call
    returned ???
 38h  WORD  returned ???
---class 05h function 0Ah---
 17h  7 BYTEs ???
 1Eh  DWORD pointer to ???
 22h  DWORD pointer to IFS open file structure (see #01661)
 26h  BYTE  seek type (02h = from end)
```

```
 28h  DWORD offset on call
    returned new absolute position
---class 05h functions 0Bh,0Ch---
 17h  7 BYTEs ???
 1Eh  DWORD pointer to ???
 22h  DWORD pointer to IFS open file structure (see #01661)
 28h  WORD  number of bytes to transfer
    returned bytes actually transferred
 2Ah  DWORD transfer address
---class 05h function 0Dh---
 17h  7 BYTEs ???
 1Eh  DWORD pointer to ???
 22h  DWORD pointer to IFS open file structure (see #01661)
 26h  BYTE  file handle???
 27h  BYTE  unused???
 28h  WORD  ???
 2Ah  WORD  ???
 2Ch  WORD  ???
 2Eh  WORD  ???
---class 05h function 0Eh---
 17h  7 BYTEs ???
 1Eh  DWORD pointer to ???
 22h  DWORD pointer to IFS open file structure (see #01661)
 26h  BYTE  00h commit file
    01h close file
 27h  BYTE  unused???
---class 05h function 0Fh---
 17h  7 BYTEs ???
 1Eh  DWORD pointer to ???
 22h  4 BYTEs ???
 26h  BYTE  02h GET attributes
    03h PUT attributes
 27h  BYTE  unused???
 28h 12 BYTEs ???
 34h  WORD  search attributes???
 36h  DWORD pointer to filename
 3Ah  WORD  (GET) returned ???
 3Ch  WORD  (GET) returned ???
 3Eh  WORD  (GET) returned ???
 40h  WORD  (GET) returned ???
 42h  WORD  (PUT) new attributes
```

```
      (GET) returned attributes
  ---class 05h function 10h---
   17h  7 BYTEs ???
   1Eh  DWORD pointer to ???
   22h  DWORD pointer to IFS open file structure (see #01661)
   26h  WORD  ???
   28h  DWORD pointer to ???
   2Ch  WORD  ???
   2Eh  BYTE  ???
   2Fh  BYTE  subfunction
      01h get printer setup
      03h ???
      04h ???
      05h ???
      06h ???
      07h ???
      21h set printer setup
  ---class 05h function 11h---
   17h  7 BYTEs ???
   1Eh  DWORD pointer to ???
   22h  DWORD pointer to IFS open file structure (see #01661)
   26h  BYTE  subfunction
   27h  BYTE  unused???
   28h  WORD  ???
   2Ah  WORD  ???
   2Ch  WORD  ???
   2Eh  BYTE  ???
   2Fh  BYTE  ???
  ---class 05h function 12h---
   17h 15 BYTEs unused???
   26h  WORD  PSP segment
   28h  BYTE  type of process termination
   29h  BYTE  unused???
  ---class 05h function 13h---
   17h 15 BYTEs unused???
   26h  WORD  PSP segment
---request class 06h---
 16h  DWORD returned pointer to string corresponding to error code at 03h
 1Ah  BYTE  returned ???
 1Bh  BYTE  unused
---request class 07h---
```

```
 16h  DWORD pointer to IFS open file structure (see #01661)
 1Ah  BYTE  ???
 1Bh  BYTE  unused???
SeeAlso: #01659,#01658,#01661

Format of IFS open file structure:
Offset  Size  Description (Table 01661)
 00h  WORD  ???
 02h  WORD  device info word
 04h  WORD  file open mode
 06h  WORD  ???
 08h  WORD  file attributes
 0Ah  WORD  owner's network machine number
 0Ch  WORD  owner's PSP segment
 0Eh  DWORD file size
 12h  DWORD current offset in file
 16h  WORD  file time
 18h  WORD  file date
 1Ah 11 BYTEs filename in FCB format
 25h  WORD  ???
 27h  WORD  hash value of SFT address
    (low word of linear address + segment&F000h)
 29h  3 WORDs network info from SFT
 2Fh  WORD  ???

Format of one item in DOS 4.0+ list of special program names:
Offset  Size  Description (Table 01662)
 00h  BYTE  length of name (00h = end of list)
 01h  N BYTEs name in format name.ext
 N    2 BYTEs DOS version to return for program (major,minor)
    (see AH=30h,INT 2F/AX=122Fh)
---DOS 4 only---
 N+2  BYTE  number of times to return fake version number (FFh = always)
Note: if the name of the executable for the program making the DOS "get
    version" call matches one of the names in this list, DOS returns the
    specified version rather than the true version number
--------v-215252-----------------------------
INT 21 - VIRUS - "516"/"Leapfrog" - INSTALLATION CHECK
  AX = 5252h
Return: BX = FFEEh if resident
SeeAlso: AX=4BFFh"Cascade",AX=58CCh
```

```
--------D-2153--------------------------------
INT 21 - DOS 2+ internal - TRANSLATE BIOS PARAMETER BLOCK TO DRIVE PARAM BLOCK
  AH = 53h
  DS:SI -> BIOS Parameter Block (see #01663)
  ES:BP -> buffer for Drive Parameter Block (see #01395 at AH=32h)
      DBP drive byte must be set to valid drive (Windows95-OSR2)
  ---Windows95---
  CX = signature 4558h ('EX') for FAT32 extended BPB/DPB (see #01664)
  DX = signature 4152h ('AR') for FAT32 extended BPB/DPB
Return: ES:BP buffer filled
Notes:  for DOS 3.0+, the cluster at which to start searching is set to 0000h
    and the number of free clusters is set to FFFFh (unknown)
  if the number of sectors per cluster is set to zero, MS-DOS will hang
    at startup because it computes the internally-used shift count by
    shifting this value right until the carry flag is set; since this
    will never happen when the field is zero, MS-DOS hangs
  not supported by Windows NT 3.1

Format of BIOS Parameter Block:
Offset  Size  Description (Table 01663)
 00h  WORD  number of bytes per sector
 02h  BYTE  number of sectors per cluster
 03h  WORD  number of reserved sectors at start of disk
 05h  BYTE  number of FATs
 06h  WORD  number of entries in root directory
 08h  WORD  total number of sectors
    for DOS 4.0+, set to zero if partition >32M, then set DWORD at
      15h to actual number of sectors
 0Ah  BYTE  media ID byte (see #01356)
 0Bh  WORD  number of sectors per FAT
---DOS 2.13---
 0Dh  WORD  number of sectors per track
 0Fh  WORD  number of heads
 11h  WORD  number of hidden sectors
---DOS 3.0+ ---
 0Dh  WORD  number of sectors per track
 0Fh  WORD  number of heads
 11h  DWORD number of hidden sectors
 15h 11 BYTEs reserved
---DOS 4.0+ ---
 15h  DWORD total number of sectors if word at 08h contains zero
```

```
 19h  6 BYTEs ???
 1Fh  WORD  number of cylinders
 21h  BYTE  device type
 22h  WORD  device attributes (removable or not, etc)
---DR DOS 5+ ---
 15h  DWORD total number of sectors if word at 08h contains zero
 19h  6 BYTEs reserved
---European MS-DOS 4.00---
 15h  DWORD total number of sectors if word at 08h contains zero
    (however, this DOS does not actually implement >32M partitions)
SeeAlso: #01395,#01664

Format of Extended BIOS Parameter Block:
Offset  Size  Description (Table 01664)
 00h 25 BYTEs same as standard DOS 4-6 BPB (see #01663)
 19h  DWORD sectors per FAT if WORD at 0Bh is 0000h
 1Dh  WORD  extended flags
    bit 7: do not mirror active FAT to inactive FATs
    bits 6-4: reserved (0)
    bits 3-0: the 0-based FAT number of the active FAT
        (if mirroring disabled)
 1Fh  WORD  file system version (high byte=major, low byte=minor)
    0000h = Win95-OSR2
 21h  DWORD starting cluster number of root directory
 25h  WORD  file system information sector number (see also #01788)
    FFFFh if none
 27h  WORD  sector number of backup boot sector (FFFFh if none)
 29h  6 WORDs reserved
SeeAlso: #01560,#01787
--------D-2154-------------------------------
INT 21 - DOS 2+ - GET VERIFY FLAG
  AH = 54h
Return: AL = verify flag
      00h off
      01h on (all disk writes verified after writing)
SeeAlso: AH=2Eh
--------v-2154--BX4475-----------------------
INT 21 - VIRUS - "Dual_GtM"/"Ganeu" - INSTALLATION CHECK
  AH = 54h
  BX = 4475h ("Du")
  CX = 616Ch ("al")
```

```
Return: BX = 4774h ("Gt") and CX = 4D21h ("M!") if resident
SeeAlso: AX=50FDh"VIRUS",AX=5454h"VIRUS"
--------v-215454----------------------------
INT 21 - VIRUS - "Dudley" - INSTALLATION CHECK
  AX = 5454h
Return: AX = 0000h if resident
SeeAlso: AX=50FDh"VIRUS",AH=54h/BX=4475h"VIRUS",AX=7BCEh"VIRUS"
--------D-2155------------------------------
INT 21 - DOS 2+ internal - CREATE CHILD PSP
  AH = 55h
  DX = segment at which to create new PSP
  SI = (DOS 3.0+) value to place in memory size field at DX:[0002h]
Return: AL destroyed
Notes:  creates a "child" PSP rather than making an exact copy of the current
    PSP; the new PSP's parent pointer is set to the current PSP and the
    reference count for each inherited file is incremented
  (DOS 2.0+) sets current PSP to DX
  (DOS 3.0+) marks "no inherit" file handles as closed in child PSP
  this function is implemented using the same code as AH=26h, so unlike
    other DOS 2+ functions, it does not return status in CF, instead
    returning status in AL as DOS 1.x functions do (but it never puts an
    explicit return value in AL)
SeeAlso: AH=26h,AH=50h
--------D-2156------------------------------
INT 21 - DOS 2+ - "RENAME" - RENAME FILE
  AH = 56h
  DS:DX -> ASCIZ filename of existing file (no wildcards, but see below)
  ES:DI -> ASCIZ new filename (no wildcards)
  CL = attribute mask (server call only, see below)
Return: CF clear if successful
  CF set on error
      AX = error code (02h,03h,05h,11h) (see #01680)
Notes:  allows move between directories on same logical volume
  this function does not set the archive attribute
    (see #01420 at AX=4301h), which results in incremental backups not
    backing up the file under its new name
  open files should not be renamed
  (DOS 2.x only) this function renames file by creating a new directory
    entry with the new name, then marking the old entry deleted
  (DOS 3.0+) allows renaming of directories
  (DOS 3.1+) wildcards are allowed if invoked via AX=5D00h, in which case
```

```
    error 12h (no more files) is returned on success, and both source and
    destination specs must be canonical (as returned by AH=60h).
    Wildcards in the destination are replaced by the corresponding char
    of each source file being renamed.  Under DOS 3.x, the call will fail
    if the destination wildcard is *.* or equivalent; under DR DOS 5.0,
    the call will fail with error code 03h if any wildcards are used.
    When invoked via AX=5D00h, only those files matching the attribute
    mask in CL are renamed.
  under the FlashTek X-32 DOS extender, the old-name pointer is in DS:EDX
    and the new-name pointer is in ES:EDI (DS must equal ES)
BUG:  (DR DOS 3.41) when invoked via AX=5D00h, this function will generate
    a new directory entry with the new name (including any wildcards)
    which can only be removed with a sector editor
SeeAlso: AH=17h,AX=4301h,AX=43FFh/BP=5053h,AX=5D00h,AH=60h,AX=7156h
SeeAlso: AX=F257h/SF=04h
--------v-215643------------------------
INT 21 - VIRUS - "PS-MPC.Gold" - INSTALLATION CHECK
  AX = 5643h  ('VC')
Return: AX = 5053h  ('PS') if resident
SeeAlso: AX=33E0h"VIRUS",AX=5741h"VIRUS",AX=6303h"VIRUS"
--------D-215700----------------------------
INT 21 - DOS 2+ - GET FILE'S LAST-WRITTEN DATE AND TIME
  AX = 5700h
  BX = file handle
Return: CF clear if successful
      CX = file's time (see #01665)
      DX = file's date (see #01666)
  CF set on error
      AX = error code (01h,06h) (see #01680)
Note: under DR DOS 3.41 and 5.0, this function returns 0 (no date/time) for
    character devices; MS-DOS returns date and time of opening
SeeAlso: AX=5701h,AX=5704h"Windows95"

Bitfields for file time:
Bit(s)  Description (Table 01665)
 15-11  hours (0-23)
 10-5 minutes
 4-0  seconds/2

Bitfields for file date:
Bit(s)  Description (Table 01666)
```

```
 15-9 year - 1980
 8-5  month
 4-0  day
--------D-215701-----------------------------
INT 21 - DOS 2+ - SET FILE'S LAST-WRITTEN DATE AND TIME
  AX = 5701h
  BX = file handle
  CX = new time (see #01665)
  DX = new date (see #01666)
Return: CF clear if successful
  CF set on error
      AX = error code (01h,06h) (see #01680)
SeeAlso: AX=5700h,AX=5705h"Windows95",AX=5707h"Windows95"
--------D-215702-----------------------------
INT 21 - DOS 4.x only - GET EXTENDED ATTRIBUTES FOR FILE
  AX = 5702h
  BX = file handle
  CX = size of result buffer or 0000h
  DS:SI -> EAP list (see #01667)
  ES:DI -> buffer for returned EAV list (see #01670)
Return: CF clear if successful
      CX = size of returned data
  CF set on error
      AX = error code (see #01680)
Desc: get the current value of one or more extended attributes
Notes:  if CX=0000h on entry, ES:DI is ignored and no data is actually
    returned, only the amount of data which is available
  the default DOS 4 behavior is to return a single word of 0000h (no
    structures) in the result buffer if CX>=0002h on entry; this
    functionality was apparently never released to the public
SeeAlso: AX=5703h,AX=5704h,AH=6Eh,INT 2F/AX=112Dh

Format of EAP (extended attribute properties) list:
Offset  Size  Description (Table 01667)
 00h  WORD  number of EAP structures following
 02h  var array of EAP structures (see #01668)
SeeAlso: #01670

Format of EAP (extended attribute property) structure:
Offset  Size  Description (Table 01668)
 00h  BYTE  attribute type
```

```
    01h boolean (either 00h or 01h)
    02h number (BYTE, WORD, or DWORD)
    03h string
    04h date stamp
    05h time stamp
 01h  WORD  EAP flags (see #01669)
 03h  BYTE  size of reference string (name)
 04h  N BYTEs reference string

Bitfields for EAP flags:
Bit(s)  Description (Table 01669)
 12 unchangeable
 13 ignore
 14 unchangeable
 15 used by COMMAND.COM for code page, but not understood by ATTRIB

Format of EAV (extended attribute value) list:
Offset  Size  Description (Table 01670)
 00h  WORD  number of EAV structures following
 02h  var array of Extended Attribute Value structures (see #01671)
SeeAlso: #01667

Format of Extended Attribute Value structures:
Offset  Size  Description (Table 01671)
 00h  4 BYTEs ???
 04h  BYTE  size of reference string
 05h  WORD  size of value
 07h  var reference string
  var value
--------O-215702-----------------------------
INT 21 - OS/2 v1.1+ Family API - DosQFileInfo
  AX = 5702h
  BX = file handle
  CX = size of buffer for information
  DX = level of information
       0001h standard file information (see #01672)
       0002h Query EA Size (see #01672)
       0003h Query EAs from List (see #01673)
       0004h Query All EAs (see #01673)
  ES:DI -> buffer for information (see #01672,#01673)
Return: CF clear if successful
```

```
  CF set on error
      AX = error code
SeeAlso: AX=5702h/BX=FFFFh,AX=5703h"OS/2",AH=6Dh"OS/2"

Format of OS/2 DosQFileInfo:
Offset  Size  Description (Table 01672)
 00h  WORD  creation date
 02h  WORD  creation time
 04h  WORD  last access date
 06h  WORD  last access time
 08h  WORD  last write date
 0Ah  WORD  last write time
 0Ch  DWORD file size in bytes
 10h  DWORD allocated space in bytes
 14h  WORD  file attributes
---level 2 only---
 16h  DWORD size of Extended Attributes in byte
SeeAlso: #01673,#01676

Format of OS/2 DosQFileInfo, EAOP structure:
Offset  Size  Description (Table 01673)
 00h  DWORD pointer to general EA list (see #01674)
 04h  DWORD pointer to buffer for full EA list, with length field set
 08h  DWORD (ret) error
Note: for info level 3, the first pointer must contain the address of a
    list of the Extended Attributes to be retrieved; for info level 4,
    it should be 0000h:0000h
SeeAlso: #01672,#01676

Format of OS/2 DosQFileInfo, General EA List:
Offset  Size  Description (Table 01674)
 00h  DWORD (call) total size of list in bytes (including this field)
    (ret) number of bytes actually used (including this field)
 04h  var Extended Attribute entries (see #01675) [packed arrray]

Format of OS/2 DosQFileInfo, General EA entry:
Offset  Size  Description (Table 01675)
 00h  BYTE  length of Extended Attribute name (excluding terminating NUL)
 01h  N BYTEs EA name
  BYTE  00h
SeeAlso: #01674
```

```
Format of OS/2 DosQFileInfo, Full EA List:
Offset  Size  Description (Table 01676)
 00h  DWORD (call) total size of list in bytes (including this field)
    (ret) number of bytes actually used (including this field)
 04h  var Extended Attribute data (see #01677) [packed array]
SeeAlso: #01672,#01673

Format of OS/2 Extended Attribute data (struct FEA):
Offset  Size  Description (Table 01677)
 00h  BYTE  flags
   bit 7: critical EA
 01h  BYTE  length of Extended Attribute name (excluding terminating NUL)
 02h  WORD  length of Extended Attribute value
 04h  N BYTEs EA name
  BYTE  00h
      M BYTEs EA value
--------O-215702BXFFFF-----------------------
INT 21 - OS/2 v1.1+ Compatibility Box Family API - DosQPathInfo
  AX = 5702h
  BX = FFFFh
  CX = size of buffer for information
  DX = level of information (0002h)
  DS:SI -> filename
  ES:DI -> buffer for FAPI path information (see #01678)
Return: CF clear if successful
      AL = 00h
  CF set on error
      AX = error code
SeeAlso: AX=5702h"OS/2",AX=5703h/BX=FFFFh

Format of FAPI path information:
Offset  Size  Description (Table 01678)
 00h 22 BYTEs ???
 16h  DWORD extended attribute size (none present if less than 5)
--------D-215703-----------------------------
INT 21 - DOS 4.x only - GET EXTENDED ATTRIBUTE PROPERTIES
  AX = 5703h
  BX = file handle
  CX = size of result buffer or 0000h
  ES:DI -> result buffer
```

```
Return: CF clear if successful
      CX = size of returned data
  CF set on error
      AX = error code (see #01680)
  ES:DI -> zero word (DOS 4.0) if CX >= 2 on entry
Desc: get a list of the extended attributes which are defined for the
    specified file
Notes:  if CX=0000h on entry, ES:DI is ignored and no data is actually
    returned, only the amount of data which is available
  the default DOS 4 behavior is to return a trivial EAP list consisting
    of the single word 0000h (no EAP structures) if CX>=0002h on entry;
    this functionality was apparently never released to the public
SeeAlso: AX=5702h,AX=5704h,AH=6Eh,INT 2F/AX=112Dh
--------O-215703-----------------------------
INT 21 - OS/2 v1.1+ Family API - DosSetFileInfo
  AX = 5703h
  BX = file handle
  CX = size of information buffer
  DX = level of information
  ES:DI -> information buffer
Return: CF clear if successful
  CF set on error
      AX = error code
SeeAlso: AX=5702h"OS/2",AX=5703h/BX=FFFFh
--------O-215703BXFFFF-----------------------
INT 21 - OS/2 v1.1+ Family API - DosSetPathInfo
  AX = 5703h
  BX = FFFFh
  CX = size of information buffer
  DX = level of information
  DS:SI -> filename
  ES:DI -> information buffer
Return: CF clear if successful
  CF set on error
      AX = error code
SeeAlso: AX=5702h/BX=FFFFh,AX=5703h"OS/2"
--------D-215704-----------------------------
INT 21 - DOS 4.x only - SET EXTENDED ATTRIBUTES
  AX = 5704h
  BX = file handle
  ES:DI -> EAV list (see #01670)
```

```
Return: CF clear if successful
  CF set on error
      AX = error code (see #01680)
Note: the default DOS 4 behavior is to do nothing and return successfully;
    this functionality was apparently never released to the public
SeeAlso: AX=5702h,AX=5703h,INT 2F/AX=112Dh
--------D-215704-----------------------------
INT 21 - MS-DOS 7/Windows95 - GET LAST ACCESS DATE AND TIME
  AX = 5704h
  BX = file handle
Return: CF clear if successful
      DX = last access date (see #01666)
      CX = last access time (currently always 0000h)
  CF set on error
      AX = error code
SeeAlso: AX=5701h,AX=5705h,AX=5706h
--------D-215705-----------------------------
INT 21 - MS-DOS 7/Windows95 - SET LAST ACCESS DATE AND TIME
  AX = 5705h
  BX = file handle
  CX = new last-access time (currently not supported, must be 0000h)
  DX = new last-access date (see #01666)
Return: CF clear if successful
  CF set on error
      AX = error code
SeeAlso: AX=5700h,AX=5704h"Windows95",AX=5707h
--------D-215706-----------------------------
INT 21 - MS-DOS 7/Windows95 - GET CREATION DATE AND TIME
  AX = 5706h
  BX = file handle
Return: CF clear if successful
      CX = creation time (see #01665)
      DX = creation date (see #01666)
      SI = number of 10-millisecond units past time in CX (0-199)
  CF set on error
      AX = error code
SeeAlso: AX=5701h,AX=5704h"Windows95",AX=5707h
--------D-215707-----------------------------
INT 21 - MS-DOS 7/Windows95 - SET CREATION DATE AND TIME
  AX = 5707h
  BX = file handle
```

```
  CX = new creation time (see #01665)
  DX = new creation date (see #01666)
  SI = new creation time: 10-millisecond units past time in CX (0-199)
Return: CF clear if successful
  CF set on error
      AX = error code
SeeAlso: AX=5700h,AX=5705h,AX=5706h
--------v-215741----------------------------
INT 21 - VIRUS - "WARP" -INSTALLATION CHECK
  AX = 5741h
Return: AX = 5250h if resident
SeeAlso: AX=5643h"VIRUS",AX=58CCh"VIRUS"
--------U-215757BX5757----------------------
INT 21 U - IBM Genie - Resident Manager - INSTALLATION CHECK
  AX = 5757h
  BX = 5757h
Return: AX = 0000h if installed
      BX = ???
      DX = ???
      DS:SI -> list of 27 DWORD entry point addresses
Program: IBM Genie is a set of utility TSRs by Helix Software
Note: other functions possible if BX <> 5757h, but details not yet available
----------215758----------------------------
INT 21 U - Headroom - API
  AX = 5758h
  BL = function
      00h ???
      01h get Headroom location
    Return: CF clear if installed
          AX = PSP segment of Headroom TSR
          BX = paragraphs of memory used by Headroom
      CF set if not (normal DOS return)
    Note: this function is also used as an installation check
      02h get INT 21 handler
    Return: CF clear
      ES:BX -> Headroom's INT 21 handler
    Note: also sets unknown flag
      03h launch application???
    DS:SI -> 233-byte application record
    Return: ???
      04h ???
```

```
???
Return: CF clear
  05h get swap directory
Return: CF clear
  DX:AX -> ASCIZ swap directory name
  06h ???
DX = ???
Return: CF clear
  07h ???
  08h ???
  09h get current application
Return: BX = application number
  0Ah ???
DX = application number
DS:SI = ???
Return: ???
  0Bh ???
  0Ch ???
DX = application number
???
Return: ???
  0Dh ???
DX = application number
???
Return: ???
  0Eh get ???
Return: CF clear
  AX = ???
  0Fh set ??? flag
  10h clear ??? flag
  11h find application by name
DS:SI -> ASCIZ application name
Return: CF clear
  AX = application number or FFFFh if not loaded
  12h ???
DX = application number
Return: CF clear
  ???
  13h ???
Return: CF clear
  14h ???
```

```
    same as function 13h
      15h set ???
    DX = ???
      16h get ???
    Return: AX = ??? set by function 15h
      17h get ???
    Return: BX = ???
      CX = ??? (may be pointer in BX:CX)
      18h BUG: branches incorrectly due to fencepost error
Program: Headroom is a TSR/task switcher by Helix Software
SeeAlso: AX=4C57h,AX=5757h,INT 2F/AX=5758h
--------D-2158--------------------------------
INT 21 - DOS 2.11+ - GET OR SET MEMORY ALLOCATION STRATEGY
  AH = 58h
  AL = subfunction
      00h get allocation strategy
    Return: AX = current strategy (see #01679)
      01h set allocation strategy
    BL = new allocation strategy (see #01679)
    BH = 00h (DOS 5+)
Return: CF clear if successful
  CF set on error
      AX = error code (01h) (see #01680)
Notes:  the Set subfunction accepts any value in BL for DOS 3.x and 4.x;
    2 or greater means last fit
  the Get subfunction returns the last value set
  setting an allocation strategy involving high memory does not
    automatically link in the UMB memory chain; this must be done
    explicitly with AX=5803h in order to actually allocate high memory
  a program which changes the allocation strategy should restore it
    before terminating
  Toshiba MS-DOS v2.11 supports subfunctions 00h and 01h, as does the
    TI Professional MS-DOS v2.13
  DR DOS 3.41 reportedly reverses subfunctions 00h and 01h
SeeAlso: AH=48h,AH=49h,AH=4Ah,INT 2F/AX=4310h"XMS",INT 67/AH=3Fh

(Table 01679)
Values for DOS memory allocation strategy:
 00h low memory first fit
 01h low memory best fit
 02h low memory last fit
```

```
---DOS 5+ ---
 40h high memory first fit
 41h high memory best fit
 42h high memory last fit
 80h first fit, try high then low memory
 81h best fit, try high then low memory
 82h last fit, try high then low memory
--------D-2158-----------------------------
INT 21 - DOS 5+ - GET OR SET UMB LINK STATE
  AH = 58h
  AL = subfunction
       02h get UMB link state
    Return: AL = current link state
          00h UMBs not part of DOS memory chain
          01h UMBs in DOS memory chain
       03h set UMB link state
    BX = new link state
        0000h remove UMBs from DOS memory chain
        0001h add UMBs to DOS memory chain
Return: CF clear if successful
  CF set on error
      AX = error code (01h) (see #01680)
Notes:  a program which changes the UMB link state should restore it before
    terminating
  UMBs will only be available if CONFIG.SYS contains the line DOS=UMB,
    the UMBs have been linked into the memory chain with AX=5803h, and
    the allocation strategy has been set to include high memory with
    AX=5801h
SeeAlso: #01687,#02766 at INT 2F/AX=4310h
--------v-2158CC-----------------------------
INT 21 - VIRUS - "1067"/"Headcrash" - INSTALLATION CHECK
  AX = 58CCh
Return: CF clear if resident
SeeAlso: AX=5741h"VIRUS",AX=5643h,AX=5252h,AX=58DDh,AX=6303h"VIRUS",AX=6969h
--------v-2158DD-----------------------------
INT 21 - VIRUS - "1067"/"Headcrash" - GET ORIGINAL INT 21h VECTOR
  AX = 58DDh
Return: CX = code segment of virus
  ES:BX = old INT 21h vector
SeeAlso: AX=5252h,AX=58CCh,AX=6969h
--------D-2159--BX0000-----------------------
```

```
INT 21 - DOS 3.0+ - GET EXTENDED ERROR INFORMATION
  AH = 59h
  BX = 0000h
Return: AX = extended error code (see #01680)
  BH = error class (see #01682)
  BL = recommended action (see #01683)
  CH = error locus (see #01684)
  ES:DI may be pointer (see #01681, #01680)
  CL, DX, SI, BP, and DS destroyed
Notes:  functions available under DOS 2.x map the true DOS 3.0+ error code into
    one supported under DOS 2.x
  you should call this function to retrieve the true error code when an
    FCB or DOS 2.x call returns an error
  under DR DOS 5.0, this function does not use any of the DOS-internal
    stacks and may thus be called at any time
SeeAlso: AH=59h/BX=0001h,AX=5D0Ah,INT 2F/AX=122Dh,INT 24

(Table 01680)
Values for DOS extended error code:
---DOS 2.0+ ---
 00h (0)   no error
 01h (1)   function number invalid
 02h (2)   file not found
 03h (3)   path not found
 04h (4)   too many open files (no handles available)
 05h (5)   access denied
 06h (6)   invalid handle
 07h (7)   memory control block destroyed
 08h (8)   insufficient memory
 09h (9)   memory block address invalid
 0Ah (10)  environment invalid (usually >32K in length)
 0Bh (11)  format invalid
 0Ch (12)  access code invalid
 0Dh (13)  data invalid
 0Eh (14)  reserved
 0Eh (14)  (PTS-DOS 6.51+, S/DOS 1.0+) fixup overflow
 0Fh (15)  invalid drive
 10h (16)  attempted to remove current directory
 11h (17)  not same device
 12h (18)  no more files
---DOS 3.0+ (INT 24 errors)---
```

```
 13h (19)  disk write-protected
 14h (20)  unknown unit
 15h (21)  drive not ready
 16h (22)  unknown command
 17h (23)  data error (CRC)
 18h (24)  bad request structure length
 19h (25)  seek error
 1Ah (26)  unknown media type (non-DOS disk)
 1Bh (27)  sector not found
 1Ch (28)  printer out of paper
 1Dh (29)  write fault
 1Eh (30)  read fault
 1Fh (31)  general failure
 20h (32)  sharing violation
 21h (33)  lock violation
 22h (34)  disk change invalid (ES:DI -> media ID structure)(see #01681)
 23h (35)  FCB unavailable
 23h (35)  (PTS-DOS 6.51+, S/DOS 1.0+) bad FAT
 24h (36)  sharing buffer overflow
 25h (37)  (DOS 4.0+) code page mismatch
 26h (38)  (DOS 4.0+) cannot complete file operation (EOF / out of input)
 27h (39)  (DOS 4.0+) insufficient disk space
 28h-31h   reserved
---OEM network errors (INT 24)---
 32h (50)  network request not supported
 33h (51)  remote computer not listening
 34h (52)  duplicate name on network
 35h (53)  network name not found
 36h (54)  network busy
 37h (55)  network device no longer exists
 38h (56)  network BIOS command limit exceeded
 39h (57)  network adapter hardware error
 3Ah (58)  incorrect response from network
 3Bh (59)  unexpected network error
 3Ch (60)  incompatible remote adapter
 3Dh (61)  print queue full
 3Eh (62)  queue not full
 3Fh (63)  not enough space to print file
 40h (64)  network name was deleted
 41h (65)  network: Access denied
    (DOS 3.0+ [maybe 3.3+???]) codepage switching not possible
```

```
      (see also INT 21/AX=6602h,INT 2F/AX=AD42h)
 42h (66)  network device type incorrect
 43h (67)  network name not found
 44h (68)  network name limit exceeded
 45h (69)  network BIOS session limit exceeded
 46h (70)  temporarily paused
 47h (71)  network request not accepted
 48h (72)  network print/disk redirection paused
 49h (73)  network software not installed
      (LANtastic) invalid network version
 4Ah (74)  unexpected adapter close
      (LANtastic) account expired
 4Bh (75)  (LANtastic) password expired
 4Ch (76)  (LANtastic) login attempt invalid at this time
 4Dh (77)  (LANtastic v3+) disk limit exceeded on network node
 4Eh (78)  (LANtastic v3+) not logged in to network node
 4Fh (79)  reserved
---end of errors reportable via INT 24---
 50h (80)  file exists
 51h (81)  (undoc) duplicated FCB
 52h (82)  cannot make directory
 53h (83)  fail on INT 24h
---network-related errors (non-INT 24)---
 54h (84)  (DOS 3.3+) too many redirections / out of structures
 55h (85)  (DOS 3.3+) duplicate redirection / already assigned
 56h (86)  (DOS 3.3+) invalid password
 57h (87)  (DOS 3.3+) invalid parameter
 58h (88)  (DOS 3.3+) network write fault
 59h (89)  (DOS 4.0+) function not supported on network / no process slots
       available
 5Ah (90)  (DOS 4.0+) required system component not installed / not frozen
 5Bh (91)  (DOS 4.0+,NetWare4) timer server table overflowed
 5Ch (92)  (DOS 4.0+,NetWare4) duplicate in timer service table
 5Dh (93)  (DOS 4.0+,NetWare4) no items to work on
 5Fh (95)  (DOS 4.0+,NetWare4) interrupted / invalid system call
 64h (100) (MSCDEX) unknown error
 64h (100) (DOS 4.0+,NetWare4) open semaphore limit exceeded
 65h (101) (MSCDEX) not ready
 65h (101) (DOS 4.0+,NetWare4) exclusive semaphore is already owned
 66h (102) (MSCDEX) EMS memory no longer valid
 66h (102) (DOS 4.0+,NetWare4) semaphore was set when close attempted
```

```
 67h (103) (MSCDEX) not High Sierra or ISO-9660 format
 67h (103) (DOS 4.0+,NetWare4) too many exclusive semaphore requests
 68h (104) (MSCDEX) door open
 68h (104) (DOS 4.0+,NetWare4) operation invalid from interrupt handler
 69h (105) (DOS 4.0+,NetWare4) semaphore owner died
 6Ah (106) (DOS 4.0+,NetWare4) semaphore limit exceeded
 6Bh (107) (DOS 4.0+,NetWare4) insert drive B: disk into A: / disk changed
 6Ch (108) (DOS 4.0+,NetWare4) drive locked by another process
 6Dh (109) (DOS 4.0+,NetWare4) broken pipe
 6Eh (110) (DOS 5.0+,NetWare4) pipe open/create failed
 6Fh (111) (DOS 5.0+,NetWare4) pipe buffer overflowed
 70h (112) (DOS 5.0+,NetWare4) disk full
 71h (113) (DOS 5.0+,NetWare4) no more search handles
 72h (114) (DOS 5.0+,NetWare4) invalid target handle for dup2
 73h (115) (DOS 5.0+,NetWare4) bad user virtual address / protection violation
 74h (116) (DOS 5.0+) VIOKBD request
 74h (116) (NetWare4) error on console I/O
 75h (117) (DOS 5.0+,NetWare4) unknown category code for IOCTL
 76h (118) (DOS 5.0+,NetWare4) invalid value for verify flag
 77h (119) (DOS 5.0+,NetWare4) level four driver not found by DOS IOCTL
 78h (120) (DOS 5.0+,NetWare4) invalid / unimplemented function number
 79h (121) (DOS 5.0+,NetWare4) semaphore timeout
 7Ah (122) (DOS 5.0+,NetWare4) buffer too small to hold return data
 7Bh (123) (DOS 5.0+,NetWare4) invalid character or bad file-system name
 7Ch (124) (DOS 5.0+,NetWare4) unimplemented information level
 7Dh (125) (DOS 5.0+,NetWare4) no volume label found
 7Eh (126) (DOS 5.0+,NetWare4) module handle not found
 7Fh (127) (DOS 5.0+,NetWare4) procedure address not found
 80h (128) (DOS 5.0+,NetWare4) CWait found no children
 81h (129) (DOS 5.0+,NetWare4) CWait children still running
 82h (130) (DOS 5.0+,NetWare4) invalid operation for direct disk-access handle
 83h (131) (DOS 5.0+,NetWare4) attempted seek to negative offset
 84h (132) (DOS 5.0+,NetWare4) attempted to seek on device or pipe
---JOIN/SUBST errors---
 85h (133) (DOS 5.0+,NetWare4) drive already has JOINed drives
 86h (134) (DOS 5.0+,NetWare4) drive is already JOINed
 87h (135) (DOS 5.0+,NetWare4) drive is already SUBSTed
 88h (136) (DOS 5.0+,NetWare4) can not delete drive which is not JOINed
 89h (137) (DOS 5.0+,NetWare4) can not delete drive which is not SUBSTed
 8Ah (138) (DOS 5.0+,NetWare4) can not JOIN to a JOINed drive
 8Bh (139) (DOS 5.0+,NetWare4) can not SUBST to a SUBSTed drive
```

```
 8Ch (140) (DOS 5.0+,NetWare4) can not JOIN to a SUBSTed drive
 8Dh (141) (DOS 5.0+,NetWare4) can not SUBST to a JOINed drive
 8Eh (142) (DOS 5.0+,NetWare4) drive is busy
 8Fh (143) (DOS 5.0+,NetWare4) can not JOIN/SUBST to same drive
 90h (144) (DOS 5.0+,NetWare4) directory must not be root directory
 91h (145) (DOS 5.0+,NetWare4) can only JOIN to empty directory
 92h (146) (DOS 5.0+,NetWare4) path is already in use for SUBST
 93h (147) (DOS 5.0+,NetWare4) path is already in use for JOIN
 94h (148) (DOS 5.0+,NetWare4) path is in use by another process
 95h (149) (DOS 5.0+,NetWare4) directory previously SUBSTituted
 96h (150) (DOS 5.0+,NetWare4) system trace error
 97h (151) (DOS 5.0+,NetWare4) invalid event count for DosMuxSemWait
 98h (152) (DOS 5.0+,NetWare4) too many waiting on mutex
 99h (153) (DOS 5.0+,NetWare4) invalid list format
 9Ah (154) (DOS 5.0+,NetWare4) volume label too large
 9Bh (155) (DOS 5.0+,NetWare4) unable to create another TCB
 9Ch (156) (DOS 5.0+,NetWare4) signal refused
 9Dh (157) (DOS 5.0+,NetWare4) segment discarded
 9Eh (158) (DOS 5.0+,NetWare4) segment not locked
 9Fh (159) (DOS 5.0+,NetWare4) invalid thread-ID address
-----
 A0h (160) (DOS 5.0+) bad arguments
 A0h (160) (NetWare4) bad environment pointer
 A1h (161) (DOS 5.0+,NetWare4) invalid pathname passed to EXEC
 A2h (162) (DOS 5.0+,NetWare4) signal already pending
 A3h (163) (DOS 5.0+) uncertain media
 A3h (163) (NetWare4) ERROR_124 mapping
 A4h (164) (DOS 5.0+) maximum number of threads reached
 A4h (164) (NetWare4) no more process slots
 A5h (165) (NetWare4) ERROR_124 mapping
 B0h (176) (MS-DOS 7.0) volume is not locked
 B1h (177) (MS-DOS 7.0) volume is locked in drive
 B2h (178) (MS-DOS 7.0) volume is not removable
 B4h (180) (MS-DOS 7.0) lock count has been exceeded
 B4h (180) (NetWare4) invalid segment number
 B5h (181) (MS-DOS 7.0) a valid eject request failed
 B5h (181) (DOS 5.0-6.0,NetWare4) invalid call gate
 B6h (182) (DOS 5.0+,NetWare4) invalid ordinal
 B7h (183) (DOS 5.0+,NetWare4) shared segment already exists
 B8h (184) (DOS 5.0+,NetWare4) no child process to wait for
 B9h (185) (DOS 5.0+,NetWare4) NoWait specified and child still running
```

```
 BAh (186) (DOS 5.0+,NetWare4) invalid flag number
 BBh (187) (DOS 5.0+,NetWare4) semaphore does not exist
 BCh (188) (DOS 5.0+,NetWare4) invalid starting code segment
 BDh (189) (DOS 5.0+,NetWare4) invalid stack segment
 BEh (190) (DOS 5.0+,NetWare4) invalid module type (DLL can not be used as
       application)
 BFh (191) (DOS 5.0+,NetWare4) invalid EXE signature
 C0h (192) (DOS 5.0+,NetWare4) EXE marked invalid
 C1h (193) (DOS 5.0+,NetWare4) bad EXE format (e.g. DOS-mode program)
 C2h (194) (DOS 5.0+,NetWare4) iterated data exceeds 64K
 C3h (195) (DOS 5.0+,NetWare4) invalid minimum allocation size
 C4h (196) (DOS 5.0+,NetWare4) dynamic link from invalid Ring
 C5h (197) (DOS 5.0+,NetWare4) IOPL not enabled
 C6h (198) (DOS 5.0+,NetWare4) invalid segment descriptor privilege level
 C7h (199) (DOS 5.0+,NetWare4) automatic data segment exceeds 64K
 C8h (200) (DOS 5.0+,NetWare4) Ring2 segment must be moveable
 C9h (201) (DOS 5.0+,NetWare4) relocation chain exceeds segment limit
 CAh (202) (DOS 5.0+,NetWare4) infinite loop in relocation chain
 CBh (203) (NetWare4) environment variable not found
 CCh (204) (NetWare4) not current country
 CDh (205) (NetWare4) no signal sent
 CEh (206) (NetWare4) file name not 8.3
 CFh (207) (NetWare4) Ring2 stack in use
 D0h (208) (NetWare4) meta expansion is too long
 D1h (209) (NetWare4) invalid signal number
 D2h (210) (NetWare4) inactive thread
 D3h (211) (NetWare4) file system information not available
 D4h (212) (NetWare4) locked error
 D5h (213) (NetWare4) attempted to execute non-family API call in DOS mode
 D6h (214) (NetWare4) too many modules
 D7h (215) (NetWare4) nesting not allowed
 E6h (230) (NetWare4) non-existent pipe, or bad operation
 E7h (231) (NetWare4) pipe is busy
 E8h (232) (NetWare4) no data available for nonblocking read
 E9h (233) (NetWare4) pipe disconnected by server
 EAh (234) (NetWare4) more data available
 FFh (255) (NetWare4) invalid drive
Note: there is a report that some Microsoft documentation shifts all DOS
    error codes in the range BCh (188) through CAh (202) up by one
    compared to what is listed here; that is probably a documentation
    error
```

```
SeeAlso: #01682,#01683,#01684,#01961

Format of media ID structure:
Offset  Size  Description (Table 01681)
 00h 12 BYTEs ASCIZ volume label of required disk
 0Ch  DWORD serial number (DOS 4.0+)
SeeAlso: #01680

(Table 01682)
Values for DOS Error Class:
 01h (1)  out of resource (storage space or I/O channels)
 02h (2)  temporary situation (file or record lock)
 03h (3)  authorization / permission problem (denied access)
 04h (4)  internal system error (system software bug)
 05h (5)  hardware failure
 06h (6)  system failure (configuration file missing or incorrect)
 07h (7)  application program error
 08h (8)  not found
 09h (9)  bad format
 0Ah (10) locked
 0Bh (11) media error
 0Ch (12) already exists / collision with existing item
 0Dh (13) unknown / other
 0Eh (14) (undoc) cannot
 0Fh (15) (undoc) time
SeeAlso: #01680,#01683,#01684

(Table 01683)
Values for DOS Suggested Action:
 01h  retry
 02h  delayed retry (after pause)
 03h  prompt user to reenter input
 04h  abort after cleanup
 05h  immediate abort ("panic")
 06h  ignore
 07h  retry after user intervention
SeeAlso: #01680,#01682,#01684

(Table 01684)
Values for DOS Error Locus:
 01h  unknown or not appropriate
```

```
 02h  block device (disk error)
 03h  network related
 04h  serial device (timeout)
 04h  (PTS-DOS 6.51+ & S/DOS 1.0+) character device
 05h  memory related
SeeAlso: #01680,#01682,#01683
--------D-2159--BX0001-----------------------
INT 21 - European MS-DOS 4.0 - GET HARD ERROR INFORMATION
  AH = 59h
  BX = 0001h
Return: ES:DI -> hard error information packet (see #01685) for most recent
    hard (critical) error
SeeAlso: AH=59h/BX=0000h,AH=95h,INT 24

Format of European MS-DOS 4.0 hard error information packet:
Offset  Size  Description (Table 01685)
 00h  WORD  contents of AX at system entry
 02h  WORD  Process ID which encountered error
 04h  WORD  contents of AX at time of error
 06h  BYTE  error type
    00h physical I/O error
    01h disk change request
    02h file sharing violation
    03h FCB problem
    04h file locking violation
    05h bad FAT
    06h network detected error
 07h  BYTE  INT 24 error code
 08h  WORD  extended error code (see #01680)
 0Ah  DWORD pointer to associated device
--------D-215A------------------------------
INT 21 - DOS 3.0+ - CREATE TEMPORARY FILE
  AH = 5Ah
  CX = file attribute (see #01420 at AX=4301h)
  DS:DX -> ASCIZ path ending with a '\' + 13 zero bytes to receive the
    generated filename
Return: CF clear if successful
      AX = file handle opened for read/write in compatibility mode
      DS:DX pathname extended with generated name for temporary file
  CF set on error
      AX = error code (03h,04h,05h) (see #01680)
```

```
Desc: creates a file with a unique name which must be explicitly deleted
Notes:  under the FlashTek X-32 DOS extender, the path pointer is in DS:EDX
  MS-DOS 3.0-4.0 and DR DOS 3.4-5.0 generate the filename as a sequence
    of hex digits based on the current date and time; MS-DOS 6+,
    DR DOS 6, and Novell DOS 7 use letters A-P in place of hex digits
  MS-DOS 5.0 and DR DOS 3.41/5.0 will insert a missing trailing backslash
    before appending the generated filename, but due to this, an empty
    string results in a file in the root directory
BUGS: COMPAQ DOS 3.31 hangs if the pathname is at XXXXh:0000h; it apparently
    wraps around to the end of the segment
  MS-DOS 5.00 revisions A and B and PC-DOS 5.00 revision A reportedly
    hang the system if the specified path is the root directory and the
    root directory is full (no free directory entries)
  Mark Incley <mincley@krisalis.demon.co.uk> reports that this function
    hangs in MS-DOS 6.2x if the name includes two consecutive path
    separators (e.g. C:\\ )
SeeAlso: AH=3Ch,AH=5Bh
--------D-215B-------------------------------
INT 21 - DOS 3.0+ - CREATE NEW FILE
  AH = 5Bh
  CX = file attribute (see #01420 at AX=4301h)
  DS:DX -> ASCIZ filename
Return: CF clear if successful
      AX = file handle opened for read/write in compatibility mode
  CF set on error
      AX = error code (03h,04h,05h,50h) (see #01680)
Notes:  unlike AH=3Ch, this function will fail if the specified file exists
    rather than truncating it; this permits its use in creating semaphore
    files because it is an atomic "test and set" operation
  under the FlashTek X-32 DOS extender, the filename pointer is in DS:EDX
SeeAlso: AH=3Ch,AH=5Ah
--------D-215C-------------------------------
INT 21 - DOS 3.0+ - "FLOCK" - RECORD LOCKING
  AH = 5Ch
  AL = subfunction
      00h lock region of file
      01h unlock region of file
  BX = file handle
  CX:DX = start offset of region within file
  SI:DI = length of region in bytes
Return: CF clear if successful
```

```
  CF set on error
      AX = error code (01h,06h,21h,24h) (see #01680)
Notes:  error returned unless SHARE or network installed
  an unlock call must specify the same region as some prior lock call
  locked regions become entirely inaccessible to other processes
  duplicate handles created with AH=45h or AH=46h inherit locks, but
    handles inherited by child processes (see AH=4Bh) do not
  under DR DOS 3.41 and 5.0, if a process opens a file without the no-
    inherit flag and then starts a child, any locks set by the parent
    are ignored, and the child will only get an error if it tries to
    lock an area previously locked by the parent process.  Under Novell
    DOS 7, this function is fully supported as in MS-DOS.
SeeAlso: AX=440Bh,AH=BCh,AH=BEh,INT 2F/AX=110Ah,INT 2F/AX=110Bh
--------D-215D00-----------------------------
INT 21 U - DOS 3.1+ internal - SERVER FUNCTION CALL
  AX = 5D00h
  DS:DX -> DOS parameter list (see #01686)
  DPL contains all register values for a call to INT 21h
Return: as appropriate for function being called
Notes:  does not check AH.  Out of range values will crash the system
  executes using specified computer ID and process ID
  sharing delay loops skipped
  a special sharing mode is enabled to handle FCBs opened across network
  wildcards are enabled for DELETE (AH=41h) and RENAME (AH=56h) under
    MS-DOS; under DR DOS 3.41, wildcards corrupt the filesystem; and
    under DR DOS 5.0-6.0, the call returns error code 03h due to improper
    support for the server function call (refer to BUGS: section below)
  an extra file attribute parameter is enabled for OPEN (AH=3Dh),
    DELETE (AH=41h), and RENAME (AH=56h)
  functions which take filenames require canonical names (as returned
    by AH=60h); this is apparently to prevent multi-hop file forwarding
BUGS: the OS/2 2.0 DOS Boot Session incorrectly maps DOS drive letters,
    seemingly ignoring HPFS drives
  DR DOS 5.0-6.0 merely recursively call INT 21 after loading the
    registers from the DPL, leading to problems for peer-to-peer
    networks
SeeAlso: AH=3Dh,AH=41h,AH=56h,AH=60h

Format of DOS parameter list:
Offset  Size  Description (Table 01686)
 00h  WORD  AX
```

```
 02h  WORD  BX
 04h  WORD  CX
 06h  WORD  DX
 08h  WORD  SI
 0Ah  WORD  DI
 0Ch  WORD  DS
 0Eh  WORD  ES
 10h  WORD  reserved (0)
 12h  WORD  computer ID (0 = current system)
 14h  WORD  process ID (PSP segment on specified computer)
Note: under Windows Enhanced mode, the computer ID is normally the virtual
    machine ID (see INT 2F/AX=1683h), though this can reportedly be
    changed by setting UniqueDOSPSP= in SYSTEM.INI
--------D-215D01-----------------------------
INT 21 U - DOS 3.1+ internal - COMMIT ALL FILES FOR SPECIFIED COMPUTER/PROCESS
  AX = 5D01h
  DS:DX -> DOS parameter list (see #01686), only computer ID and
      process ID fields used
Return: CF set on error
      AX = error code (see #01680)
  CF clear if successful
Notes:  flushes buffers and updates directory entries for each file which has
    been written to; if remote file, calls INT 2F/AX=1107h
  the computer ID and process ID are stored but ignored under DOS 3.3
  not supported by DR DOS 3.41 and 5.0; returns error code 01h.  Fully
    supported by Novell DOS 7
SeeAlso: AH=0Dh,AH=68h,INT 2F/AX=1107h
--------D-215D02-----------------------------
INT 21 U - DOS 3.1+ internal - SHARE.EXE - CLOSE FILE BY NAME
  AX = 5D02h
  DS:DX -> DOS parameter list (see #01686), only fields DX, DS,
      computer ID, and process ID used
  DPL's DS:DX -> ASCIZ name of file to close
Return: CF set on error
      AX = error code (see #01680)
  CF clear if successful
Notes:  error unless SHARE is loaded (calls [SysFileTable-28h])
    (see #01636 at AH=52h)
  name must be canonical fully-qualified, such as returned by AH=60h
  not supported by DR DOS 3.41 and 5.0; returns error code 01h
  not supported by Novell DOS 7, although in Update 15 it is reported
```

```
    to no longer return an error code
SeeAlso: AX=5D03h,AX=5D04h,AH=3Eh,AH=60h
--------D-215D03-----------------------------
INT 21 U - DOS 3.1+ internal - SHARE.EXE - CLOSE ALL FILES FOR GIVEN COMPUTER
  AX = 5D03h
  DS:DX -> DOS parameter list (see #01686), only computer ID used
Return: CF set on error
      AX = error code (see #01680)
  CF clear if successful
Notes:  error unless SHARE is loaded (calls [SysFileTable-30h])
    (see #01636 at AH=52h)
  not supported by DR DOS 3.41 and 5.0; returns error code 01h
  not supported by Novell DOS 7 (at least through Update 4; may be
    supported in Update 15)
SeeAlso: AX=5D02h,AX=5D04h
--------D-215D04-----------------------------
INT 21 U - DOS 3.1+ internal - SHARE.EXE - CLOSE ALL FILES FOR GIVEN PROCESS
  AX = 5D04h
  DS:DX -> DOS parameter list (see #01686), only computer ID and
      process ID fields used
Return: CF set on error
      AX = error code (see #01680)
  CF clear if successful
Notes:  error unless SHARE is loaded (calls [SysFileTable-2Ch])
     (see #01636 at AH=52h)
  not supported by DR DOS 3.41 and 5.0; returns error code 01h
  not supported by Novell DOS 7
SeeAlso: AX=5D02h,AX=5D03h,INT 2F/AX=111Dh
--------D-215D05-----------------------------
INT 21 U - DOS 3.1+ internal - SHARE.EXE - GET OPEN FILE LIST ENTRY
  AX = 5D05h
  DS:DX -> DOS parameter list (see #01686)
  DPL's BX = index of sharing record (see #01637 at AH=52h)
  DPL's CX = index of SFT in sharing record's SFT list
Return: CF clear if successful
      ES:DI -> ASCIZ filename
      BX = network machine number of SFT's owner
      CX = number of locks held by SFT's owner
  CF set if either index out of range
      AX = 0012h (no more files)
Notes:  error unless SHARE is loaded (calls [SysFileTable-18h])
```

```
    (see #01636 at AH=52h)
  names are always canonical fully-qualified, such as returned by AH=60h
  not supported by DR DOS 3.41 and 5.0 and Novell DOS 7, but does not
    return an error, instead destroying AX
SeeAlso: AH=5Ch,AH=60h
--------D-215D06----------------------------
INT 21 U - DOS 3.0+ internal - GET ADDRESS OF DOS SWAPPABLE DATA AREA
  AX = 5D06h
Return: CF set on error
     AX = error code (see #01680)
  CF clear if successful
      DS:SI -> nonreentrant data area (includes all three DOS stacks)
    (critical error flag is first byte) (see #01687)
      CX = size in bytes of area which must be swapped while in DOS
      DX = size in bytes of area which must always be swapped
Notes:  the Critical Error flag is used in conjunction with the InDOS flag
    (see AH=34h) to determine when it is safe to enter DOS from a TSR
  setting CritErr flag allows use of functions 50h/51h from INT 28h under
    DOS 2.x by forcing use of correct stack
  swapping the data area allows reentering DOS unless DOS is in a
    critical section delimited by INT 2A/AH=80h and INT 2A/AH=81h,82h
  under DOS 4.0, AX=5D0Bh should be used instead of this function
  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.
  DR DOS 3.41+ supports this function, but the SDA format beyond the
    first 18h bytes is completely different from MS-DOS
BUG:  calling this function with certain values in DX crashes Novell DOS 7.0
    prior to Update 14
SeeAlso: AX=5D0Bh,INT 2A/AH=80h,INT 2A/AH=81h,INT 2A/AH=82h

Format of DOS 3.10-3.30 Swappable Data Area:
Offset  Size  Description (Table 01687)
 -34  BYTE  (DOS 3.10+) printer echo flag (00h off, FFh active)
 -31  BYTE  (DOS 3.30) current switch character
 -30  BYTE  current memory allocation strategy (see AH=58h)
 -28  BYTE  (DOS 3.30) incremented on each INT 21/AX=5E01h call
 -27 16 BYTEs (DOS 3.30) 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)
 -1 BYTE  unused padding
```

```
---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 sets to drive number during INT 24, 00h otherwise)
 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
---remainder need only be swapped if in DOS---
 18h  WORD  value of AX on call to INT 21
 1Ah  WORD  PSP segment for sharing/network
 1Ch  WORD  network machine number for sharing/network (0000h = us)
 1Eh  WORD  first usable memory block found when allocating memory
 20h  WORD  best usable memory block found when allocating memory
 22h  WORD  last usable memory block found when allocating memory
 24h  WORD  memory size in paragraphs (used only during initialization)
 26h  WORD  last entry checked during directory search
 28h  BYTE  flag: INT 24 returned Fail
 29h  BYTE  flags: allowable INT 24 actions (passed to INT 24 in AH)
 2Ah  BYTE  directory flag (00h directory, 01h file)
 2Bh  BYTE  flag: FFh if Ctrl-Break termination, 00h otherwise
 2Ch  BYTE  flag: allow embedded blanks in FCB
 2Dh  BYTE  padding (unused)
 2Eh  BYTE  day of month
 2Fh  BYTE  month
 30h  WORD  year - 1980
 32h  WORD  number of days since 01jan1980
 34h  BYTE  day of week (0 = Sunday)
```

```
 35h  BYTE  flag: console swapped during read from device
 36h  BYTE  flag: safe to call INT 28 if nonzero
 37h  BYTE  flag: if nonzero, INT 24 Abort turned into INT 24 Fail
    (set only during process termination)
 38h 26 BYTEs device driver request header (see #02597 at INT 2F/AX=0802h)
 52h  DWORD pointer to device driver entry point (used in calling driver)
 56h 22 BYTEs device driver request header for I/O calls
 6Ch 14 BYTEs device driver request header for disk status check
 7Ah  DWORD pointer to device I/O buffer???
 7Eh  WORD  ???
 80h  WORD  ???
 82h  BYTE  type of PSP copy (00h=simple for INT 21/AH=26h, FFh=make child)
 83h  BYTE  padding (unused)
 84h  3 BYTEs 24-bit user number (see AH=30h)
 87h  BYTE  OEM number (see #01394 at AH=30h)
 88h  WORD  offset to error code conversion table for INT 25/INT 26
 8Ah  6 BYTEs CLOCK$ transfer record (see #01688)
 90h  BYTE  device I/O buffer for single-byte I/O functions
 91h  BYTE  padding??? (unused)
 92h 128 BYTEs  buffer for filename
112h 128 BYTEs  buffer for filename
192h 21 BYTEs findfirst/findnext search data block (see #01626 at AH=4Eh)
1A7h 32 BYTEs directory entry for found file (see #01352 at AH=11h)
1C7h 81 BYTEs copy of current directory structure for drive being accessed
218h 11 BYTEs FCB-format filename for device name comparison
223h  BYTE  terminating NUL for above filename
224h 11 BYTEs wildcard destination specification for rename (FCB format)
22Fh  BYTE  terminating NUL for above spec
230h  BYTE  ???
231h  WORD  destination file/directory starting sector
233h  5 BYTEs ???
238h  BYTE  extended FCB file attribute
239h  BYTE  type of FCB (00h regular, FFh extended)
23Ah  BYTE  directory search attributes
23Bh  BYTE  file open/access mode
23Ch  BYTE  file found/delete flag
    bit 0: file found
    bit 4: file deleted
23Dh  BYTE  flag: device name found on rename, or file not found
23Eh  BYTE  splice flag (file name and directory name together)
23Fh  BYTE  flag indicating how DOS function was invoked
```

```
    (00h = direct INT 20/INT 21, FFh = server call AX=5D00h)
240h  BYTE  sector position within cluster
241h  BYTE  flag: translate sector/cluster (00h no, 01h yes)
242h  BYTE  flag: 00h if read, 01h if write
243h  BYTE  current working drive number
244h  BYTE  cluster factor
245h  BYTE  flag: cluster split mode
246h  BYTE  line edit (AH=0Ah) insert mode flag (nonzero = on)
247h  BYTE  canonicalized filename referred to existing file/dir if FFh
248h  BYTE  volume ID flag
249h  BYTE  type of process termination (00h-03h) (see AH=4Dh)
24Ah  BYTE  file create flag (00h = no, search only)
24Bh  BYTE  value with which to replace first byte of deleted file's name
      (normally E5h, but 00h as described under INT 21/AH=13h)
24Ch  DWORD pointer to Drive Parameter Block for critical error invocation
    temp: used during process termination
250h  DWORD pointer to stack frame containing user registers on INT 21
254h  WORD  stores SP across INT 24
256h  DWORD pointer to DOS Drive Parameter Block for ???
25Ah  WORD  saving partial cluster number
25Ch  WORD  temp: sector of work current cluster
25Eh  WORD  high part of cluster number (only low byte referenced)
260h  WORD  ??? temp
262h  BYTE  Media ID byte returned by AH=1Bh,1Ch
263h  BYTE  padding (unused)
264h  DWORD pointer to device header when filename is character device
268h  DWORD pointer to current SFT
26Ch  DWORD pointer to current directory structure for drive being accessed
270h  DWORD pointer to caller's FCB
274h  WORD  number of SFT to which file being opened will refer
276h  WORD  temporary storage for file handle
278h  DWORD pointer to a JFT entry in process handle table
      (see #01378 at AH=26h)
27Ch  WORD  offset in DOS DS of first filename argument
27Eh  WORD  offset in DOS DS of second filename argument
280h  WORD  offset of last component in pathname or FFFFh
282h  WORD  offset of transfer address to add
284h  WORD  last relative cluster within file being accessed
286h  WORD  temp: absolute cluster number being accessed
288h  WORD  directory sector number
28Ah  WORD  ??? current cluster number
```

```
28Ch  WORD  current relative sector number within file
28Eh  WORD  current sector number
290h  WORD  current byte offset within sector
292h  DWORD current offset in file
296h  DWORD temp: file byte count
29Ah  WORD  temp: file byte count
29Ch  WORD  free file cluster entry
29Eh  WORD  last file cluster entry
2A0h  WORD  next file cluster number
2A2h  DWORD number of bytes appended to file
2A6h  DWORD pointer to current work disk buffer
2AAh  DWORD pointer to working SFT
2AEh  WORD  used by INT 21 dispatcher to store caller's BX
2B0h  WORD  used by INT 21 dispatcher to store caller's DS
2B2h  WORD  temporary storage while saving/restoring caller's registers
2B4h  DWORD pointer to prev call frame (offset 250h) if INT 21 reentered
    also switched to for duration of INT 24
2B8h 21 BYTEs FindFirst search data for source file(s) of a rename operation
    (see #01626 at AH=4Eh)
2CDh 32 BYTEs directory entry for file being renamed (see #01352 at AH=11h)
2EDh 331 BYTEs  critical error stack
   403h  35 BYTEs scratch SFT
438h 384 BYTEs  disk stack (functions greater than 0Ch, INT 25,INT 26)
5B8h 384 BYTEs  character I/O stack (functions 01h through 0Ch)
---DOS 3.2,3.3x only---
738h  BYTE  device driver lookahead flag (usually printer)
    (see AH=64h"DOS 3.2+")
739h  BYTE  volume change flag
73Ah  BYTE  flag: virtual open
73Bh  BYTE  ???
Note: the only fields which remain valid BETWEEN calls to INT 21h are those
    in the initial "swap-always" portion of the SDA
SeeAlso: #01690

Format of CLOCK$ transfer record:
Offset  Size  Description (Table 01688)
 00h  WORD  number of days since 1-Jan-1980
 02h  BYTE  minutes
 03h  BYTE  hours
 04h  BYTE  hundredths of second
 05h  BYTE  seconds
```

```
--------D-215D07-----------------------------
INT 21 U - DOS 3.1+ network - GET REDIRECTED PRINTER MODE
  AX = 5D07h
Return: DL = mode
      00h redirected output is combined
      01h redirected output in separate print jobs
Note: MS-DOS and DR DOS 3.41/5.0 simply push AX on the stack and call
    INT 2F/AX=1125h
SeeAlso: AX=5D08h,AX=5D09h,INT 2F/AX=1125h
--------D-215D08-----------------------------
INT 21 U - DOS 3.1+ network - SET REDIRECTED PRINTER MODE
  AX = 5D08h
  DL = mode
      00h redirected output is combined
      01h redirected output placed in separate jobs, start new print job
    now
Return: nothing
Note: MS-DOS and DR DOS 3.41/5.0 simply push AX on the stack and call
    INT 2F/AX=1125h
SeeAlso: AX=5D07h,AX=5D09h,INT 2F/AX=1125h
--------D-215D09-----------------------------
INT 21 U - DOS 3.1+ network - FLUSH REDIRECTED PRINTER OUTPUT
  AX = 5D09h
Return: nothing
Desc: forces redirected printer output to be printed, and starts a new print
    job
Notes:  MS-DOS and DR DOS 3.41/5.0 simply push AX on the stack and call
    INT 2F/AX=1125h
  this function is also supported by 10Net, which calls it Terminate All
    Spool Jobs, and does not flush if in "combine" mode
SeeAlso: AX=5D07h,AX=5D08h,INT 2F/AX=1125h
--------D-215D0A-----------------------------
INT 21 - DOS 3.1+ - SET EXTENDED ERROR INFORMATION
  AX = 5D0Ah
  DS:DX -> 11-word DOS parameter list (see #01686)
Return: nothing.  next call to AH=59h will return values from fields AX,BX,CX,
    DX,DI, and ES in corresponding registers
Notes:  documented for DOS 5+, but undocumented in earlier versions
  the MS-DOS Programmer's Reference incorrectly states that this call was
    introduced in DOS 4, and fails to mention that the ERROR structure
    passed to this function is a DOS parameter list.
```

```
BUGS: DR DOS 3.41 and 5.0 read the value for ES from the DS field of the
    DPL; fortunately, MS-DOS ignores the DS field, allowing a generic
    routine which sets both DS and ES fields to the same value
  Novell DOS 7 (through at least Update 4) does not save the pointer,
    which is always reported as 0000h:0000h by AH=59h; Update 15 fully
    supports this function
SeeAlso: AH=59h/BX=0000h
--------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
--------!---Section-------------------------
```