```
Interrupt List, part 14 of 18
Copyright (c) 1989-1999,2000 Ralf Brown
--------F-2FDA00-----------------------------
INT 2F - ZyXEL ZFAX v1.x - INSTALLATION CHECK
  AX = DA00h
Return: AH = enabled state (00h = enabled, 01h = disabled)
  AL = 5Ah installed
Program: ZFAX is the bundled FAX software which comes with the ZyXEL model
    fax modems.
Note: This function, and the other DAxxh functions, may apply only to version
    1 of the software; see AX=DB00h for the version 2 installation check
SeeAlso: AX=CBDCh,AX=DA01h,AX=DA02h,AX=DA03h,AX=DB00h
--------F-2FDA01-----------------------------
INT 2F - ZyXEL ZFAX v1.x - UNINSTALL
  AX = DA01h
Return: AL = 00h Success
       01h Failure
SeeAlso: AX=DA00h,AX=DB01h
--------F-2FDA02-----------------------------
INT 2F - ZyXEL ZFAX v.1x - DISABLE
  AX = DA02h
Return: AL = 00h
SeeAlso: AX=DA03h,AX=DB02h
--------F-2FDA03-----------------------------
INT 2F - ZyXEL ZFAX v1.x - ENABLE
  AX = DA03h
Return: AL = 00h
SeeAlso: AX=DA02h,AX=DB03h
--------G-2FDA55-----------------------------
INT 2F U - TRAP.COM - INSTALLATION CHECK
  AX = DA55h
  DL = interrupt number
  DH = ???
Return: if installed
      AH = interrupt number
      AL = ???
      ES:BX -> ???
Program: TRAP is an interrupt call tracer by Patrick Phillipot/Udo Chrosziel
Note: a separate copy of TRAP is loaded for each interrupt to be traced; thus
    the interrupt number is part of the installation check
--------N-2FDAB2-----------------------------
```

```
INT 2F U - Beame&Whiteside BWSNMP - INSTALLATION CHECK
  AX = DAB2h
Return: AX = 00FFh if installed
      BX:CX -> MIB table
Program: BWSNMP is part of the BW-NFS package
SeeAlso: INT 62/AH=00h"ETHDEV"
--------F-2FDB00-----------------------------
INT 2F - ZyXEL ZFAX v2+ - INSTALLATION CHECK
  AX = DB00h
Return: AL = 5Bh if installed (v2.x)
      ES:BX -> configuration table???
  AX = 00DBh if installed (v3)
      ES:BX -> ZFAX configuration table (see #03092)
Program: ZFAX is the bundled FAX software which comes with the ZyXEL model
    Fax modems.
SeeAlso: AX=CBDCh,AX=DA00h,AX=DB01h,AX=DB02h,AX=DB03h

Format of ZFAX Configuration Table:
Offset  Size  Description (Table 03092)
 00h  WORD  table version number (0300h for v3.0-v4.01)
 02h  BYTE  reserved
 03h 70 BYTEs ZFAX working path
 49h 128 BYTEs  path to external editor
 C9h 128 BYTEs  path to external terminal emulator
149h 128 BYTEs  path to Ring Shell
1C9h 128 BYTEs  path to DOS Shell
249h 128 BYTEs  path to Data Shell
2C9h 70 BYTEs path to Chinese font
30Fh  BYTE  printer type (see #03093)
310h  BYTE  printer port (00h = LPT1, etc.)
311h  BYTE  type of graphics adapter
    (00h auto-detect, 01h VGA, 02h EGA, 03h CGA, 04h Hercules)
312h  BYTE  display type
    (00h auto-detect, 01h LCD, 02h color, 03h mono)
313h  BYTE  scan code for ZFAX hotkey (see #00006)
314h  BYTE  shift mask for ZFAX hotkey
315h  BYTE  tone/pulse dialing (00h tone, 01h pulse)
316h  BYTE  Caller ID (00h disabled, 01h enabled)
317h  BYTE  Distinctive Ring (00h disabled, 01h enabled)
318h  BYTE  normal ring answer type (see #03094)
319h  BYTE  Ring 1 answer type (see #03094)
```

```
31Ah  BYTE  Ring 2 answer type (see #03094)
31Bh  BYTE  Ring 3 answer type (see #03094)
31Ch  BYTE  COM port for modem
31Dh  BYTE  speaker volume (00h-07h)
31Eh  BYTE  modem dial timer, seconds
31Fh  WORD  user-defined COM port I/O address
321h  BYTE  user-defined COM port IRQ number
322h 81 BYTEs dial prefix string
373h 79 BYTEs dial postfix string
3C2h  BYTE  ring count until automatic answer
3C3h  BYTE  retry count on busy signal
3C4h  BYTE  redial delay in seconds
3C5h  WORD  system password
3C7h  BYTE  reserved
3C8h  BYTE  voice file compression format
    00h CELP at 9600bps
    01h two-bit ADPCM at 19200bps
    02h three-bit ADPCM at 28800bps
3C9h  BYTE  voice system: DTMF 0 action (see #03095)
3CAh  BYTE  voice system: DTMF 1 action
3CBh  BYTE  voice system: DTMF 2 action
3CCh  BYTE  voice system: DTMF 3 action
3CDh  BYTE  voice system: DTMF 4 action
3CEh  BYTE  voice system: DTMF 5 action
3CFh  BYTE  voice system: DTMF 6 action
3D0h  BYTE  voice system: DTMF 7 action
3D1h  BYTE  voice system: DTMF 8 action
3D2h  BYTE  voice system: DTMF 9 action (see #03095)
3D3h  WORD  reserved
3D5h 25 BYTEs local FAX ID to display on page header
3EEh 20 BYTEs local FAX ID sent to remote FAX
402h  BYTE  FAX page size
    00h A4 (210x297mm)
    01h B4 (250x353mm)
    02h A3 (297x420mm)
403h  BYTE  FAX resolution
    00h normal (3.85 pixels/mm)
    01h high (7.7 pixels/mm)
404h  BYTE  FAX coding scheme
    00h 1-D, modified Huffman coding
    01h 2-D, modified READ coding
```

```
405h  BYTE  left margin for text in millimeters
406h  BYTE  vertical insertion for text in mm (0-20)
407h  BYTE  horizontal insertion in mm (0-20)
408h  BYTE  maximum text lines per page
409h  BYTE  text type (00h ASCII, 01h WordStar-formatted)
40Ah  BYTE  PCX image resize (00h disabled, 01h enabled)
40Bh  BYTE  AutoPrint (00h disabled, 01h enabled)
40Ch  BYTE  cover page (00h disabled, 01h enabled)
40Dh 81 BYTEs cover page logo filename
45Eh 65 BYTEs cover page sender name
49Fh  BYTE  print capture (00h disabled, 01h enabled)
4A0h  BYTE  send immediately (00h disabled, 01h enabled)
4A1h  BYTE  print capture printer port
4A2h  BYTE  print capture timer in seconds
4A3h  BYTE  scan code for print capture hotkey (see #00006)
4A4h  BYTE  shift mask for print capture hotkey
4A5h  BYTE  DataShell type
    00h internal Zmodem, 01h Data Shell, 02h disable
4A6h  BYTE  video I/O type
    00h auto-detect, 01h use BIOS, 02h direct writes
4A7h  BYTE  call transfer digits (0-9)
4A8h  WORD  voice recorder maximum time in seconds (0-999)

(Table 03093)
Values for ZFAX printer type:
 00h  EPSON FX (9 pins)
 01h  EPSON LQ (24 pins)
 02h  HP Laser Jet II, letter size
 03h  HP Laser Jet II, legal size
 04h  HP Laser Jet II, A4 size
 05h  HP Laser Jet III, letter size
 06h  HP Laser Jet III, legal size
 07h  HP Laser Jet III, A4 size
SeeAlso: #03092

(Table 03094)
Values for Ring Answer Type:
 00h  voice system
 01h  FAX only
 02h  Data Shell
 03h  Ring Shell
```

```
 04h  DOS Shell
 05h  ignore
SeeAlso: #03092

(Table 03095)
Values for DTMF action:
 00h  none
 01h  page operator
 02h  FaxBack
 03h  announcement
 04h  call transfer
 05h  receive FAX
 06h  receive data
 07h  voice mailbox
 08h  DOS Shell Out
 09h  Data Shell Out
SeeAlso: #03092
--------U-2FDB00-----------------------------
INT 2F U - WINGO.COM - INSTALLATION CHECK
  AX = DB00h
Return: AX = FFFFh if installed
  CX = 5749h ('WI') if installed
  DX = 4E47h ('NG') if installed
      ES = segment of resident code
Program: WINGO.COM is TSR for starting Windows programs from DOS
    prompt (together with companion Windows program
    WINSTART.EXE) written by Douglas Boling, contributing editor of
    PC Magazine
Range:  AH=DBh to AH=FFh, selected by scanning for a free multiplex number
SeeAlso: AX=DB01h"WINGO",AX=DB02h"WINGO",AX=DB03h"WINGO",AX=DB04h"WINGO"
SeeAlso: AX=DB05h"WINGO",AX=DB06h"WINGO"
--------F-2FDB01-----------------------------
INT 2F - ZyXEL ZFAX v2+ - UNINSTALL
  AX = DB01h
Return: AX = status
      0000h successful
      0001h ZFAX is busy
      0002h another program resident above ZFAX
Note: this function unhooks the vectors taken by the ZFAX TSR if they have
    not been hooked by other TSRs and releases the TSR's memory
  ZFAX v2.x crashes the contributor's machine when this function is
```

```
    called
SeeAlso: AX=DA01h,AX=DB00h"ZFAX",AX=DB02h"ZFAX"
--------U-2FDB01-----------------------------
INT 2F U - WINGO.COM - SET CALLBACK FUNCTION POINTER
  AX = DB01h
  CX:DX -> new callback function
Return: AX = 0000h
  CX:DX -> end of resident code (stack ???)
Desc: the callback function is used for WINGO <-> WINSTART communication
SeeAlso: AX=DB00h"WINGO",AX=DB02h"WINGO",AX=DB03h"WINGO",AX=DB04h"WINGO"
SeeAlso: AX=DB05h"WINGO",AX=DB06h"WINGO"
--------F-2FDB02-----------------------------
INT 2F - ZyXEL ZFAX v2.x - DISABLE
  AX = DB02h
Return: AL = 00h
SeeAlso: AX=DA02h,AX=DB00h"ZFAX",AX=DB01h"ZFAX",AX=DB03h"ZFAX"
--------U-2FDB02-----------------------------
INT 2F U - WINGO.COM - RESET CALLBACK FUNCTION POINTER
  AX = DB02h
Return: AX = 0000h
Desc: the callback function is used for WINGO <-> WINSTART communication
SeeAlso: AX=DB00h"WINGO",AX=DB01h"WINGO",AX=DB03h"WINGO",AX=DB04h"WINGO"
SeeAlso: AX=DB05h"WINGO",AX=DB06h"WINGO"
--------F-2FDB03-----------------------------
INT 2F - ZyXEL ZFAX v2.x - ENABLE
  AX = DB03h
Return: AL = 00h
SeeAlso: AX=DA03h,AX=DB00h"ZFAX",AX=DB02h"ZFAX"
--------U-2FDB03-----------------------------
INT 2F U - WINGO.COM - SET ??? FLAG
  AX = DB03h
Return: AX = 0000h
SeeAlso: AX=DB00h"WINGO",AX=DB04h"WINGO",AX=DB05h"WINGO",AX=DB06h"WINGO"
--------U-2FDB04-----------------------------
INT 2F U - WINGO.COM - RESET ??? FLAG
  AX = DB04h
Return: AX = 0000h
SeeAlso: AX=DB00h"WINGO",AX=DB03h"WINGO",AX=DB05h"WINGO",AX=DB06h"WINGO"
--------U-2FDB05-----------------------------
INT 2F U - WINGO.COM - GET ??? FLAG
  AX = DB05h
```

```
Return: AX = 0000h
  DX = unknown flag - 0 or 1
Program: WINGO.COM is TSR for starting Windows programs from DOS
    prompt (together with companion Windows program
    WINSTART.EXE) written by Douglas Boling, contributing editor of
    PC Magazine
SeeAlso: AX=DB00h"WINGO",AX=DB03h"WINGO",AX=DB04h"WINGO"
--------U-2FDB06-----------------------------
INT 2F U - WINGO.COM - CALL CALLBACK FUNCTION
  AX = DB06h
Return: AX = return value of INT 2F/AX=1685h
Note: uses Windows service INT 2F/AX=1685h (SWITCH VMs AND CALLBACK) to
    call the callback function previously set by AX=DB01h
  used for WINGO <-> WINSTART communication
SeeAlso: AX=1685h, AX=DB00h"WINGO",AX=DB01h"WINGO",AX=DB02h"WINGO",
SeeAlso: AX=DB03h"WINGO",AX=DB04h"WINGO",AX=DB05h"WINGO"
--------F-2FDB10-----------------------------
INT 2F - ZyXEL ZFAX v3+ - EXECUTE ZFAX MAIN MENU
  AX = DB10h
SeeAlso: AX=DB00h"ZFAX",AX=DB11h"ZFAX"
--------F-2FDB11-----------------------------
INT 2F - ZyXEL ZFAX v3+ - SEND FAX
  AX = DB11h
  DS:SI -> filename including path (max 128 characters)
  DS:BX -> remote FAX number
Return: AX = status (see #03096)
SeeAlso: AX=DB00h"ZFAX",AX=DB12h,AX=DB13h,AX=DB14h,AX=DB15h,AX=DB20h,AX=DB21h

(Table 03096)
Values for ZFAX status:
 00h  OK
 01h  invalid DOS function
 02h  file not found
 03h  path not found
 04h  no file handle available
 05h  access denied by DOS
 06h  invalid handle
 07h  disk full
 10h  printer error
 11h  no graphics font
 12h  no ZFAX font
```

```
 20h  DCD dropped while sending
 21h  not ZyXEL modem
 22h  busy
 23h  no response from COM port
 24h  no carrier
 25h  no dial tone
 26h  no answer
 27h  no response
 28h  failed to send FAX
 30h  user aborted
 40h  critical error on disk
 50h  parameter error
--------F-2FDB12-----------------------------
INT 2F - ZyXEL ZFAX v3+ - PRINT FAX
  AX = DB12h
  DS:SI -> filename, including path (max 128 characters)
Return: AX = status (see #03096)
SeeAlso: AX=DB11h,AX=DB13h,AX=DB14h
--------F-2FDB13-----------------------------
INT 2F - ZyXEL ZFAX v3+ - CONVERT FAX
  AX = DB13h
  DS:SI -> source filename, including path (max 128 characters)
  DS:BX -> destination filename, including path (max 80 characters)
  CX = destination file format
       00h FAX, 01h PCX, 02h TIFF, 03h PRN
Return: AX = status (see #03096)
SeeAlso: AX=DB11h,AX=DB12h,AX=DB14h,AX=DB22h
--------F-2FDB14-----------------------------
INT 2F - ZyXEL ZFAX v3+ - VIEW FAX
  AX = DB14h
  DS:SI -> source filename, including path (max 128 characters)
Return: AX = status (see #03096)
SeeAlso: AX=DB11h,AX=DB12h,AX=DB14h
--------F-2FDB15-----------------------------
INT 2F - ZyXEL ZFAX v4.01 - SEND FAX WITH LOGO AND SIGNATURE
  AX = DB15h
  DS:SI -> source filename, including path (max 128 characters)
  DS:BX -> remote FAX number
  DS:CX -> Logo filename
  DS:DX -> Signature filename
Return: AX = status (see #03096)
```

```
SeeAlso: AX=DB00h"ZFAX",AX=DB11h
--------F-2FDB16-----------------------------
INT 2F - ZyXEL ZFAX v4.01 - QUOTE PREVIOUS STATUS
  AX = DB16h
Return: DX:AX -> previous receive state transaction log (see #03097)
  BX = type (00h incoming FAX, 01h incoming data, 02h incoming voice)
SeeAlso: AX=DB00h"ZFAX",AX=DB11h,AX=DB17h

Format of ZFAX transaction log receive state:
Offset  Size  Description (Table 03097)
 00h  WORD  year
 02h  WORD  date
 04h  WORD  time
 06h  WORD  extension number
 08h  WORD  type: 00h incoming FAX, 01h outgoing FAX, 02h voice message
 0Ah  WORD  status: 00h normal, 01h remote delete
 0Ch 125 BYTEs  filename
 89h 24 BYTEs caller ID
 A1h 20 BYTEs FAX number
 B5h 20 BYTEs remote FAX ID
 C9h  WORD  error code
 CBh  WORD  connection direction (00h transmit, 01h receive)
 CDh  BYTE  connection time minutes
 CEh  BYTE  connection time seconds
 CFh  BYTE  ???
 D0h  WORD  connection speed (24, 48, 72, 96, 12, 144)
 D2h  WORD  connection page size (A4, B4, A3)
 D4h  WORD  connection coding scheme (1-DN, 1-DH, 2-DN, 2-DH)
 D6h  WORD  page count
--------F-2FDB17-----------------------------
INT 2F - ZyXEL ZFAX v4.01 - BEGIN RECEIVING INCOMING CALL
  AX = DB17h
Return: AX = status (0000h successful, 0001h unable to start at this time)
SeeAlso: AX=DB00h"ZFAX"
--------F-2FDB18-----------------------------
INT 2F - ZyXEL ZFAX v4.01 - SCHEDULE CALL
  AX = DB18h
  DS:SI -> schedule data (see #03097)
Return: AX = status (see #03096)
SeeAlso: AX=DB00h"ZFAX"
--------F-2FDB20-----------------------------
```

```
INT 2F - ZyXEL ZFAX v3+ - POLL FAX
  AX = DB20h
  DS:SI -> remote FAX number
Return: AX = status (see #03096)
SeeAlso: AX=DB11h,AX=DB21h
--------F-2FDB21----------------------------
INT 2F - ZyXEL ZFAX v3+ - SEND VOICE
  AX = DB21h
  DS:SI -> filename including path (max 128 characters)
  DS:BX -> remote phone number
Return: AX = status (see #03096)
SeeAlso: AX=DB11h,AX=DB20h,AX=DB22h,AX=DB23h,AX=DB24h
--------F-2FDB22----------------------------
INT 2F - ZyXEL ZFAX v3+ - CONVERT VOICE FILE
  AX = DB22h
  DS:SI -> source filename, including path (max 128 characters)
  DS:BX -> destination filename, including path (max 80 characters)
  CX = destination format
       00h two-bit ADPCM, 01h three-bit ADPCM, 02h VOC
Return: AX = status (see #03096)
SeeAlso: AX=DB13h,AX=DB20h,AX=DB21h,AX=DB23h
--------F-2FDB23----------------------------
INT 2F - ZyXEL ZFAX v3+ - RECORD VOICE FILE
  AX = DB23h
  DS:SI -> destination filename, including path (max 128 characters)
  CX = recording channel (0 = telephone line, 1 = microphone/speaker)
  DX = voice file format
       00h CELP, 01h two-bit ADPCM, 02h three-bit ADPCM
Return: AX = status (see #03096)
SeeAlso: AX=DB21h,AX=DB22h,AX=DB24h
--------F-2FDB24----------------------------
INT 2F - ZyXEL ZFAX v3+ - PLAY VOICE FILE
  AX = DB24h
  DS:SI -> name of voice file, including path (max 128 characters)
  CX = playback channel (0 = telephone line, 1 = microphone/speaker)
Return: AX = status (see #03096)
SeeAlso: AX=DB21h,AX=DB22h,AX=DB24h
--------F-2FDB25----------------------------
INT 2F - ZyXEL ZFAX v3+ - ANSWER MODEM WITH VOICE SYSTEM
  AX = DB25h
Return: AX = status (see #03096)
```

```
SeeAlso: AX=DB21h,AX=DB22h,AX=DB26h,AX=DB27h,AX=DB28h
--------F-2FDB26-----------------------------
INT 2F - ZyXEL ZFAX v3+ - DIAL PHONE
  AX = DB26h
  DS:SI -> remote phone number
Return: AX = status (see #03096)
SeeAlso: AX=DB25h
--------F-2FDB27-----------------------------
INT 2F - ZyXEL ZFAX v3+ - RECEIVE FAX - ANSWER MODEM AND SET TO FAX MODE
  AX = DB27h
Return: AX = status (see #03096)
SeeAlso: AX=DB25h,AX=DB28h
--------F-2FDB28-----------------------------
INT 2F - ZyXEL ZFAX v3+ - RECEIVE FAX DATA - ANSWER MODEM IN MULTI-AUDIO MODE
  AX = DB28h
Return: AX = status (see #03096)
SeeAlso: AX=DB25h,AX=DB27h,AX=DB41h
--------F-2FDB40-----------------------------
INT 2F - ZyXEL ZFAX v3+ - INTERNAL TERMINAL
  AX = DB40h
Return: AX = status (see #03096)
SeeAlso: AX=DB00h
--------F-2FDB41-----------------------------
INT 2F - ZyXEL ZFAX v3+ - SEND DATA - DIAL NUMBER AND UPLOAD FILE WITH ZMODEM
  AX = DB41h
  DS:SI -> source file name, including path (max 128 characters)
  DS:BX -> remote data number
Return: AX = status (see #03096)
SeeAlso: AX=DB00h,AX=DB28h
--------K-2FDC00-----------------------------
INT 2F - GOLD.COM - INSTALLATION CHECK
  AX = DC00h
Return: AL = state
      00h not installed
      FFh installed
Program: GOLD is a TSR by Bob Eager which makes the NumLock key return the
    code for F1; the purpose is to improve Kermit's VTxxx emulation
--------K-2FDC01-----------------------------
INT 2F - GOLD.COM - GET STATE
  AX = DC01h
Return: AL = status
```

```
      00h off
      01h on
SeeAlso: AX=DC00h,AX=DC02h
--------K-2FDC02-----------------------------
INT 2F - GOLD.COM - SET STATE
  AX = DC02h
  DL = new state
      00h off
      01h on
Return: AL = 00h (OK)
SeeAlso: AX=DC01h
--------t-2FDD------------------------------
INT 2F - CappaCom programs - API
  AH = DDh
  AL = 00h general installation check
      Return: AL = FFh if any CappaCom programs are resident
  AL = FEh get info
      Return: ES:BX -> TSR info list (see #03098)
  AL = program identifier
      BH = function
    FDh get version
        Return: BX = version
    FFh installation check
        Return: AL = FFh if installed
        BX = version
        ES = segment of resident code
    others vary by program
Return: AL = status
      bit 7 set on error
      AL = 81h unknown function
Note: CappaCom was originally SoftCom but changed its name due to a trademark
    conflict
Index:  installation check;SoftCom programs
Index:  installation check;CappaCom programs

Format of CappaCom TSR info list:
Offset  Size  Description (Table 03098)
 00h  9 BYTEs blank-padded ASCIZ program name
 09h  BYTE  program ID
 0Ah  WORD  program's PSP segment
 0Ch  WORD  program version (major in high byte)
```

```
 0Eh  DWORD pointer to next item in info list or 0000h:0000h
 12h  BYTE  number of interrupts hooked
 13h  5 BYTEs interrupt numbers hooked by program
 18h  8 BYTEs reserved
----------2FDD--------------------------------
INT 2F - MIXFIX.EXE - API
  AH = DDh
  AL = function
       00h installation check
    Return: AX = 00DDh if installed
          BX = version (BH = major, BL = minor)
      41h/61h get From: address
    Return: AX = 0001h
      ES:BX -> ASCIZ 4d address of mail sender ("1:2/3.4")
      49h/69h get To: address
    Return: AX = 0001h
      ES:BX -> ASCIZ 4d address of recipient ("1:2/3.4")
      4Ah/6Ah get subject of mail
    Return: AX = 0001h
      ES:BX -> ASCIZ subject of handled mail
      4Dh/6Dh get mail name
    Return: AX = 0001h
      ES:BX -> ASCIZ full name of current mail file
      4Eh/6Eh get From: field
    Return: AX = 0001h
      ES:BX -> ASCIZ From: field of mail (mail sender's name)
Program: MIXFIX by "KIV without Co" is a FidoNet mail robot which may execute
    other programs for mail handling.  The called programs may use the
    services described here to retrieve information about the mail being
    handled.
Index: installation check;MIXFIX.EXE
--------d-2FDD--BX7844-----------------------
INT 2F - xDISK v3.32+ - INSTALLATION CHECK
  AH = DDh
  BX = 7844h ('xD')
  CX = 4953h ('IS')
  DX = 4B3Fh ('K?')
  AL = desired drive (01h-1Ah) or 00h to check for xDISK on any drive
  ES:DI -> 25-byte data buffer (see #03099)
Return: AX = DDFFh if installed (on specified drive if AL nonzero on entry)
     BX = 87BBh
```

```
     DX = B4C0h
     ES:DI buffer filled
  CX,CF destroyed
SeeAlso: INT 21/AX=4404h"xDISK",INT 21/AX=4405h"xDISK"

Format of xDISK data buffer:
Offset  Size  Description (Table 03099)
 00h  DWORD pointer to ASCIZ driver signature "xDISK unit: X"
 04h  BYTE  flag: 01h if disk linked to DOS, 00h if unlinked
 05h  BYTE  flag: 01h if write protected, 00h if not
 06h  BYTE  flag: 01h if root directory full, 00h if not
 07h  BYTE  flag: 01h if free space uncompacted, 00h if compacted
 08h  BYTE  resizing state: 00h not resizable, 01h resized, 80h resizable
 09h  BYTE  flag: 01h inelastic resizable disk, 00h elastic
 0Ah  2 BYTEs reserved
 0Ch  BYTE  flag: 01h collapsed disk, 00h not collapsed
 0Dh  BYTE  flag: 01h using all EMS, 00h some EMS free
 0Eh  BYTE  flag: 01h password enabled, 00h disabled
 0Fh  BYTE  flag: 01h password audio feedback, 00h no feedback
 10h  BYTE  flag: 01h password video feedback, 00h no feedback
 11h  BYTE  flag: 01h confirm changes, 00h no confirmation
 12h  BYTE  flag: 01h terse display, 00h verbose display
 13h  BYTE  flag: 01h click speaker on disk access, 00h no click
 14h  BYTE  flag: 01h flash icon on disk access, 00h no icon flash
 15h  BYTE  FAT entry size: 00h 12-bit, FFh 16-bit
 16h  WORD  count of open files in RAM disk
 18h  BYTE  unused
--------N-2FDE00BL00-------------------------
INT 2F U - Novell Netware - RPRINTER, NPRINTER - INSTALLATION CHECK
  AX = DE00h
  BL = 00h
Return: AL = FFh If Rprinter/Nprinter Installed
  BX -> Program Segment Prefix N/Rprinter.exe
--------Q-2FDE00BX4456-----------------------
INT 2F - DESQview v2.26+ External Device Interface - INSTALLATION CHECK
  AX = DE00h
  BX = 4456h ("DV")
  CX = 5844h ("XD")
  DX = 4931h ("I1")
Return: AL = FFh if installed (even if other registers do not match)
  if BX,CX, and DX were as specified on entry,
```

```
      BX = 4845h ("HE")
      CX = 5245h ("RE")
      DX = 4456h ("DV")
Range:  AH=C0h to AH=FFh, selected by scanning AH=DEh-FFh, then AH=C0h-DDh
Note: the XDI handler should not issue any DOS or BIOS calls, nor should it
    issue DESQview API calls other than those allowed from hardware ints
SeeAlso: AX=DE02h,INT 15/AX=5400h
--------Q-2FDE01-----------------------------
INT 2F - DESQview v2.26+ External Device Interface - DRIVER CUSTOM SUBFUNCTION
  AX = DE01h
  BX = driver ID
  other registers as needed by driver
Notes:  XDI drivers should pass this call through to previous handler if ID
    does not match
  DESQview never calls this function
--------Q-2FDE01BX4450-----------------------
INT 2F U - Quarterdeck QDPMI.SYS v1.0 - INSTALLATION CHECK
  AX = DE01h
  BX = 4450h ("DP")
  CX = 4D49h ("MI")
  DX = 3039h ("09")
Return: AL = FFh if installed
      BX = 4D42h ("MB")
      CX = 4921h ("I!")
      DX = 8F4Fh
      ES:DI -> filename of DPMI host overlay
InstallCheck: test for the existence of the character device QDPMI$$$
SeeAlso: INT 2F/AX=1687h,INT 31/AX=0000h
Index:  installation check;QDPMI
--------U-2FDE01BX5242-----------------------
INT 2F - DESQview v2.26+ XDI - CUSTOM SUBFUNCTION, Ralf Brown's XDI drivers
  AX = DE01h
  BX = 5242h ("RB")
  CX:DX = program identifier
    656F7000h ("eop",0) for DVeop
Return: AX = 5242h ("RB") if installed
      ES:BX -> data or entry point (see #03100)
      CX = version number (CH = major, CL = minor)

(Table 03100)
Call DVeop entry point with:
```

```
  ES:DI -> callback address or 0000h:0000h to remove callback
Return: AX = status
      0000h failed (callback table full or attempted to remove non-
      existent callback)
      0001h successful
    ES:DI -> chaining address
  BX,CX,DX destroyed
Notes:  the callback function is called with a simulated interrupt when the
    DESQview window containing it is closed; it should perform all
    necessary cleanup and then perform a FAR jump to the chaining address
    or an IRET if the chaining address is 0000h:0000h
  if the program wishes to remove itself before the window is closed, it
    should call the DVeop entry point with the previously returned
    chaining address and ignore the returned chaining address.
--------U-2FDE01BX7474-----------------------
INT 2F - DESQview v2.26+ XDI - CUSTOM SUBFUNCTION, DVTXDI.COM
  AX = DE01h
  BX = 7474h
  CL = function
      00h installation check
    Return: AL = FFh
      01h get process handle
    DX = keys on Open Window menu (DL = first, DH = second)
    Return: AX = process handle or 0000h if not running
      02h (v1.3+) set TMAN handle
    DX = TMAN process handle
      03h (v1.3+) set open keys to ignore on next CL=01h call
    DX = keys on Open Window menu (DL = first, DH = second)
Return: BX = 4F4Bh ("OK")
  DL destroyed
Note: DVTXDI is distributed as part of the shareware products DVTree (DOS
   shell/DESQview process manager) and DVTMAN by Mike Weaver
Index:  installation check;DVTXDI
--------U-2FDE01BX7575-----------------------
INT 2F - DESQview v2.26+ XDI - CUSTOM SUBFUNCTION, DVSIXDI.COM
  AX = DE01h
  BX = 7575h
  CX = function
      0000h installation check
    Return: AX = 00FFh if installed
      0001h turn on notification (currently unused)
```

```
    Return: AX = 0001h
      0002h turn off notification (currently unused)
    Return: AX = 0001h
      0003h get process information
    Return: AX = status
          0000h failed
          0001h successful
        BX = last instantaneous time slice
            in 1/100s (v1.10)
            in 1/18s (v1.11+)
        CX = number of processes
        DX = number of "(starting)" records (v2.00+)
        SI = number of records in process info array
              (v2.00+) (always 15 for v1.x)
        ES:DI -> process info array (see #03101,#03102)
      0004h get version
    Return: AH = major version
      AL = minor version
      0005h (v1.10+) get time since DESQview started
    Return: DX:AX = 1/100s since DV start (v1.10)
      DX:AX = 1/18s since DV start (v1.11+)
      0006h (v1.10+) get number of task switches
    Return: DX:AX = total task switches
      CX = task switches in last instantaneous interval
Notes:  DVSIXDI is part of the DVSI (DESQview System Information) package by
    Daniel J. Bodoh
  for v1.00, function 0003h allocates common memory, which the caller
    must deallocate after reading the process information; only the
    currently used records are placed in the buffer
  for v1.10+, function 0003h merely returns a pointer to the internal
    array of process information; the caller should make a copy of the
    array while inside a critical section (see INT 15/AX=101Bh).  Only
    those records with bit 7 of the first byte set are valid.
Index:  installation check;DVSIXDI

Format of DVSIXDI v1.00 information for one process:
Offset  Size  Description (Table 03101)
 00h  BYTE  flags
    bit 7: process slot is valid
 01h  WORD  offset into DESQVIEW.DVO of program's record if started from
      Open Windows menu, else undefined
```

```
 03h  WORD  Switch Windows window number
 05h  WORD  segment of process handle
 07h  WORD  number of tasks owned by process
 09h  WORD  mapping context of process (see #00416 at INT 15/AX=1016h)
 0Bh  DWORD hook for other programs

Format of DVSIXDI v1.10-v2.00 information for one process:
Offset  Size  Description (Table 03102)
 00h  BYTE  process flags (see #03103)
 01h  WORD  Open Window keys
 03h  WORD  Switch Windows number
 05h  WORD  segment of process handle
 07h  WORD  number of tasks for process
 09h  WORD  process mapping context
 0Bh  DWORD time process started (relative to start of DESQview)
 0Fh  DWORD time process last got CPU (relative to start of DESQview)
 13h  DWORD time process last gave up CPU (relative to start of DESQview)
 17h  DWORD total CPU time since process started
 1Bh  DWORD CPU time at start of current instantaneous interval
 1Fh  DWORD CPU time in current instantaneous interval
 23h  DWORD hook for other programs
Note: all times are in 1/100s for v1.10, in 1/18s for v1.11+

Bitfields for DVSIXDI process flags:
Bit(s)  Description (Table 03103)
 7  valid record
 6  (v2.00+) record is allocated; if bit 7 clear, process is "(starting)"
    and only offsets 01h and 09h are valid
 5  (v2.00+) this app currently owns the CPU
 4  reserved (0)
 3  DESQview system task
 2  reserved (0)
 1  task has keyboard (currently unused)
 0  task swapped out (currently unused)
--------Q-2FDE01BXFFFE-----------------------
INT 2F U - DESQview v2.26+ XDI - DVXMS.DVR - ???
  AX = DE01h
  BX = FFFEh
  CX = 4D47h ("MG")
  DX = 0052h (0,"R")
Return: AL = FFh
```

```
  DX = 584Dh
--------Q-2FDE02-----------------------------
INT 2F C - DESQview v2.26+ External Dev Interface - DV INITIALIZATION COMPLETE
  AX = DE02h
  BX = mapping context of DESQview
  DX = handle of DESQview system task
Note: driver should pass this call to previous handler after doing its work
SeeAlso: AX=DE03h,AX=DE0Fh,INT 15/AX=5400h
--------Q-2FDE03-----------------------------
INT 2F C - DESQview v2.26+ External Dev Interface - DV TERMINATION
  AX = DE03h
  BX = mapping context of DESQview
  DX = handle of DESQview system task
Notes:  driver should pass this call to previous handler before doing its work
  DESQview makes this call when it is exiting, but before unhooking any
    interrupt vectors
SeeAlso: AX=DE02h,AX=DE0Fh,INT 15/AX=5407h
--------Q-2FDE04-----------------------------
INT 2F C - DESQview v2.26+ External Dev Interface - ADD PROCESS
  AX = DE04h
  BX = mapping context of new process (see #00416 at INT 15/AX=1016h)
  DX = handle of process
Return: nothing
Notes:  XMS XDI handler (installed by default) allocates a 22-byte record
    (see #03104) from "common" memory to control access to XMS memory
  all DOS, BIOS, and DV API calls are valid in handler
  driver should pass this call to previous handler after processing it
SeeAlso: AX=DE05h,AX=DE06h,INT 15/AX=5401h

Format of XMS XDI structure:
Offset  Size  Description (Table 03104)
 00h  DWORD pointer to 10-byte record???
 04h  DWORD pointer to next XMS XDI structure
 08h  WORD  mapping context
 0Ah  BYTE  ???
 0Bh  5 BYTEs XMS entry point to return for INT 2F/AX=4310h"XMS"
    (FAR jump to next field)
 10h  6 BYTEs FAR handler for XMS driver entry point
    (consists of a FAR CALL followed by RETF)
--------Q-2FDE05-----------------------------
INT 2F C - DESQview v2.26+ External Dev Interface - REMOVE PROCESS
```

```
  AX = DE05h
  BX = mapping context of process (see #00416 at INT 15/AX=1016h)
  DX = handle of last task in process
Return: nothing
Notes:  XMS XDI handler releases the structure allocated by AX=DE04h
  driver should pass this call to previous handler before processing it
  all DOS, BIOS, and DV API calls except those generating a task switch
    are valid in handler
SeeAlso: AX=DE04h,AX=DE07h,INT 15/AX=5402h
--------Q-2FDE06----------------------------
INT 2F C - DESQview v2.26+ External Dev Interface - CREATE TASK
  AX = DE06h
  BX = mapping context of process containing task
  DX = handle of new task
Notes:  driver should pass this call to previous handler after processing it
  all DOS, BIOS, and DV API calls are valid in handler
--------Q-2FDE07----------------------------
INT 2F C - DESQview v2.26+ External Dev Interface - TERMINATE TASK
  AX = DE07h
  BX = mapping context of process containing task
  DX = handle of task
Notes:  driver should pass this call to previous handler before processing it
  all DOS, BIOS, and DV API calls except those generating a task switch
    are valid in handler
SeeAlso: AX=DE04h,AX=DE06h,AX=DE10h
--------Q-2FDE08----------------------------
INT 2F C - DESQview v2.26+ External Dev Interface - SAVE STATE
  AX = DE08h
  BX = mapping context of task being switched from
       (see #00416 at INT 15/AX=1016h)
  DX = handle of task being switched from
Notes:  invoked prior to task swap, interrupts, etc
  driver should pass this call to previous handler after processing it
SeeAlso: AX=DE09h,INT 15/AX=5403h,INT 15/AX=DE27h
--------Q-2FDE09----------------------------
INT 2F C - DESQview v2.26+ External Dev Interface - RESTORE STATE
  AX = DE09h
  BX = mapping context of task being switched to
       (see #00416 at INT 15/AX=1016h)
  DX = handle of task being switched to
Notes:  state is restored except for interrupts
```

```
  driver should pass this call to previous handler before processing it
SeeAlso: AX=DE08h,INT 15/AX=5404h,INT 15/AX=DE27h
--------Q-2FDE0A-----------------------------
INT 2F C - DESQview v2.26+ External Dev Interface - CHANGE KEYBOARD FOCUS
  AX = DE0Ah
  BX = mapping context of task receiving focus
  DX = handle of running task
Notes:  driver should pass this call to previous handler before processing it
  this call often occurs inside a keyboard interrupt
  DV 2.42 does not provide this call to XDI handlers running inside a
    window; instead, it directly calls the INT 2F handler which was
    active at the time DV started
SeeAlso: INT 15/AX=DE26h,INT 15/AX=DE2Fh
--------Q-2FDE0B-----------------------------
INT 2F C - DESQview v2.26+ External Dev Interface - DVP PROCESSING COMPLETE
  AX = DE0Bh
  BX = mapping context of DESQview system task
  CX = number of system memory paragraphs required for the use of all
       XDI drivers (DV will add this to system memory in DVP buffer)
  DX = handle of DESQview system task
  SI = mapping context of new process if it starts
  ES:DI -> DVP buffer
Return: CX incremented as needed
Notes:  once DV invokes this function, the DVP buffer contents may be changed
  driver should pass this call to previous handler before processing it
--------Q-2FDE0C-----------------------------
INT 2F C - DESQview v2.26+ External Dev Interface - SWAP OUT PROCESS
  AX = DE0Ch
  BX = mapping context of task being swapped out
       (see #00416 at INT 15/AX=1016h)
  DX = handle of DESQview system task
Note: driver should pass this call to previous handler after processing it
--------Q-2FDE0D-----------------------------
INT 2F C - DESQview v2.26+ External Dev Interface - SWAP IN PROCESS
  AX = DE0Dh
  BX = mapping context of process just swapped in
       (see #00416 at INT 15/AX=1016h)
  DX = handle of DESQview system task
Note: driver should pass this call to previous handler before processing it
--------Q-2FDE0E-----------------------------
INT 2F C - DESQview v2.26+ External Dev Interface - DVP START FAILED
```

```
  AX = DE0Eh
  BX = mapping context of DESQview system task
  DX = handle of DESQview system task
  SI = mapping context of failed process (same as for call to AX=DE0Bh)
Note: driver should pass this call to previous handler after processing it
--------Q-2FDE0F-----------------------------
INT 2F C - DESQview v2.50+ External Dev Interface - INITIALIZE DV
  AX = DE0Fh
Note: DESQview 2.50+ calls this function just before it completes its
    initialization.  At the time of the call, DESQview has not yet
    changed any interrupt vectors
SeeAlso: AX=DE02h
--------Q-2FDE10-----------------------------
INT 2F C - DESQview v2.50+ External Dev Interface - FREE TASK
  AX = DE10h
  BX = mapping context of process (see #00416 at INT 15/AX=1016h)
  DX = task handle of process
Note: DESQview 2.50+ calls this function before it frees the task; it is
    similar to AX=DE07h but allows the XDI handler to make calls which
    cause context switches
SeeAlso: AX=DE06h,AX=DE07h
--------c-2FDF00-----------------------------
INT 2F - HyperWare programs - INSTALLATION CHECK
  AX = DF00h
  BX = product code (see #03105)
  CX = 0000h
  DX = 0000h
Return: AL = status
      00h not installed
      FFh multiplex number in use
    CX = 5948h ('YH') if selected product installed
    ---HyperDisk---
    BX = code segment of resident portion
    DX = HyperDisk local data version
Program: HyperDisk is a shareware disk cache by HyperWare (Roger Cross)
Range:  AH=C0h to AH=FFh, selected by scanning AH=DFh, then AH=C0h-FFh
SeeAlso: INT 13/AX=8EEDh
Index:  installation check;HyperDisk|installation check;HyperStb
Index:  installation check;HyperKey|installation check;HyperScreen
Index:  HyperDisk;installation check|HyperStb;installation check
Index:  HyperKey;installation check|HyperScreen;installation check
```

```
(Table 03105)
Values for HyperWare product code:
 4248h  ('BH') HyperStb
 4448h  ('DH') HyperDisk v4.20+
 4B48h  ('KH') HyperKey
 5348h  ('SH') HyperScreen
--------Q-2FDF00BX5445-----------------------
INT 2F U - Quarterdeck TELTSR.COM - INSTALLATION CHECK
  AX = DF00h
  BX = 5445h ('TE')
  CX = 4C54h ('LT')
  DX = 5352h ('SR')
Return: BX = 5454h ('TT') if installed
  CX = 494Eh ('IN') if installed
  DX = 5454h ('ST') if installed
Program: TELTSR is a Telnet TSR included with Quarterdeck's DESQview/X v2.00
    which provides an INT 14h interface to the network
Range:  AH=DEh to AH=FFh and AH=C0h to AH=DDh, selected by scanning
SeeAlso: AX=DF01h"TELTSR",AX=DF02h"TELTSR",INT 14/AH=56h
--------N-2FDF01BX0000-----------------------
INT 2F U - MSG.COM - INSTALLATION CHECK
  AX = DF01h
  BX = 0000h
Return: AX = FDFFh if installed
     BX = segment of resident code
Program: MSG.COM is a TSR for intercepting incoming Novell broadcast
    messages written by Gary Dobbins (Dobbins@Arizona.Edu)
--------c-2FDF01BX4448-----------------------
INT 2F - HyperDisk v4.50+ - GET CURRENT CACHE STATE
  AX = DF01h
  BX = 4448h ('DH')
Return: AX = 0000h if function supported
      BX = number of cache buffers in use
      CX = number of cache buffers which have been modified
      DL = caching flags (see #03106)
Range:  AH=C0h to AH=FFh, selected by scanning AH=DFh, then AH=C0h-FFh
SeeAlso: AX=DF00h,AX=DF02h

Bitfields for HyperDisk caching flags:
Bit(s)  Description (Table 03106)
```

```
 0  staged writes enabled for floppy disks
 1  staged writes enabled for hard disks
 2  writes verified on floppy disks
 3  writes verified on hard disks
 4  reserved (0)
 5  reserved (0)
 6  floppy caching enabled
 7  all caching functions enabled
--------Q-2FDF01-----------------------------
INT 2F U - Quarterdeck TELTSR.COM - CLOSE CONNECTION
  AX = DF01h
Return: nothing???
Note: invokes the DESQview/X socket API (see INT 15/AX=DE2Eh) function 0006h
    to close the socket corresponding to the file handle set with
    AX=DF02h; NOP if no file handle was ever set
SeeAlso: AX=DF00h"TELTSR",AX=DF02h"TELTSR"
--------c-2FDF02BX4448-----------------------
INT 2F - HyperDisk v4.50+ - SET CACHE STATE
  AX = DF02h
  BX = 4448h ('DH')
  DL = new caching flags (see #03106)
Return: AX = 0000h if supported
      BX = number of cache buffers in use
      CX = number of cache buffers which have been modified
      DL = previous caching flags (see #03106)
Program: HyperDisk is a shareware disk cache by HyperWare (Roger Cross)
Range:  AH=C0h to AH=FFh, selected by scanning AH=DFh, then AH=C0h-FFh
SeeAlso: AX=DF00h,AX=DF01h
--------Q-2FDF02-----------------------------
INT 2F U - Quarterdeck TELTSR.COM - OPEN CONNECTION
  AX = DF02h
  BX = file handle for connection's socket
Return: AX = status
      0000h successful
      FFFFh no more room in TELTSR's JFT
Program: TELTSR is a Telnet TSR included with Quarterdeck's DESQview/X v2.00
Notes:  the indicated file handle becomes owned by TELTSR, and is closed in
    the caller's JFT
  multiple calls to this function will override the previous assignment
    without closing the previous file; use AX=DF01h before further calls
SeeAlso: AX=DF00h"TELTSR",AX=DF01h"TELTSR"
```

```
--------U-2FE000-----------------------------
INT 2F - SETDRVER.COM v2.10+ - INSTALLATION CHECK
  AX = E000h
Return: AX = 4A52h ("JR") if present
Program: SETDRVER is a public domain TSR by Jacob Rieper which sets the
    apparent DOS version analogously to MS-DOS SETVER
Notes:  this installation check differs from the usual one of returning AL=FFh
  the SETDRVER API is fully emulated by Matthias Paul's FREEVER
SeeAlso: AX=E000h/DX=4D50h,AX=E001h,INT 21/AH=52h
--------j-2FE000-----------------------------
INT 2F - KAOSHIDE - INSTALLATION CHECK
  AX = E000h
Return: AL = FFh if installed
Program: KAOSHIDE ('Hidden Kaos') is a PD joke TSR which randomly capitalizes
    alphabetic keys, written by Philip Maland. Although not a virus,
    v2.0 uses some viral-like techniques to hide itself from
    memory-reports by reducing DOS memory size.
--------U-2FE000DX4D50-----------------------
INT 2F - FREEVER v1.0+ - INSTALLATION CHECK
  AX = E000h
  DX = 4D50h ('MP')
Return: AL = FFh if installed
      AH = AMIS INT 2Dh multiplex ID
      CX = FREEVER version (CH=major, CL=minor)
      DX:DI -> AMIS-compliant signature (see #02569)
    (vendor ID is "M. Paul ", program name is "FREEVER ")
Program: FREEVER is an AMIS-conformant freeware DOS version-faking TSR similar
    to SETVER for any DOS-compatible OS, written by Matthias Paul
Note: FREEVER emulates INT 21/AH=30h, INT 21/AX=3306h, INT 21/AX=4412h,
    INT 21/AX=4452h, and SETDRVER's API on INT 2F/AX=E00xh, as well as
    providing an AMIS API on INT 2Dh
SeeAlso: AX=E000h"SETDRVER",INT 21/AH=30h,INT 21/AX=3306h,INT 21/AX=4452h
--------K-2FE000DX5354-----------------------
INT 2F - StuffIt v3.21+ - INSTALLATION CHECK
  AX = E000h
  DX = 5354h ("ST")
Return: AL = FFh if installed
      BX = version (BH = major, BL = BCD minor)
      DX = segment of resident code
Program: StuffIt is a freeware delayed keyboard stuffer by Terje Mathisen
--------U-2FE001-----------------------------
```

```
INT 2F - SETDRVER.COM v2.10+ - GET SETDRVER VERSION
  AX = E001h
Return: AH = major version
  AL = minor version
Note: the SETDRVER API is fully emulated by Matthias Paul's FREEVER, which
    returns version 2.10 for this call; use AX=E000h/DX=45D0h to get
    FREEVER's version number
SeeAlso: AX=E000h,AX=E000h/DX=45D0h
--------U-2FE002-----------------------------
INT 2F - SETDRVER.COM v2.10+ - GET ORIGINAL DOS VERSION INFO
  AX = E002h
Return: AL = FFh if successful
      BH = major DOS version
      BL = minor DOS version
      CH = DOS version flag
      CL = OEM number
      DH = major DR DOS version number (FFh if unknown)
      DL = minor DR DOS version number (FFh if unknown)
SeeAlso: AX=E003h,AX=E007h,INT 21/AH=30h
--------U-2FE003-----------------------------
INT 2F - SETDRVER.COM v2.10+ - RESET INTERNAL VARIABLES
  AX = E003h
  BH = new major DOS version
  BL = new minor DOS version
  CH = new DOS version flag
  CL = new DOS revision number
  DH = new OEM number
SeeAlso: AX=E002h
--------U-2FE004-----------------------------
INT 2F - SETDRVER.COM v2.10+ - ENABLE TSR
  AX = E004h
Return: AL = FFh if successful
SeeAlso: AX=E000h,AX=E005h,AX=E006h
--------U-2FE005-----------------------------
INT 2F - SETDRVER.COM v2.10+ - DISABLE TSR
  AX = E005h
Return: AL = FFh if successful
SeeAlso: AX=E000h,AX=E004h,AX=E006h
--------U-2FE006-----------------------------
INT 2F - SETDRVER.COM v2.10+ - GET TSR STATUS
  AX = E006h
```

```
Return: AL = FFh if successful
      BL = status
    01h resident and active
    02h resident and inactive
--------U-2FE007-----------------------------
INT 2F - SETDRVER.COM v2.10+ - GET TaskMAX STATUS AT INSTALLATION
  AX = E007h
Return: AL = FFh if successful
      BL = status
    00h if TaskMAX not loaded before SETDRVER
    FFh if TaskMAX was loaded before SETDRVER
SeeAlso: AX=E003h
--------U-2FE0-------------------------------
INT 2F - SETDRVER.COM - RESERVED FOR FUTURE USE
  AH = E0h
  AL = 08h-10h
--------K-2FE100-----------------------------
INT 2F - Phantom2 v1.1+ - INSTALLATION CHECK
  AX = E100h
Return: AX = 0001h if installed
      DS:SI -> ASCIZ hotkey name
      DS:DI -> ASCIZ recording filename
Program: Phantom of the Keyboard II is a shareware keystroke recorder/replayer
    by P2 Enterprises
SeeAlso: AX=E101h,AX=E102h,AX=E103h,AX=E300h
Index:  hotkeys;Phantom2
--------K-2FE101-----------------------------
INT 2F - Phantom2 v1.1+ - FUNCTION REQUEST
  AX = E101h
  BX = function mask (see #03107)
  CX = code for hotkey (as returned by INT 16/AH=00h) if BX bit 6 set
  DS:DX -> ASCIZ filespec if BX bit 7 set
SeeAlso: AX=E100h
Index:  hotkeys;Phantom2

Bitfields for Phantom2 function mask:
Bit(s)  Description (Table 03107)
 0  record
 1  play
 2  QuickPlay
 3  loop
```

```
 4  mode display toggle
 5  sound toggle
 6  set hotkey
 7  set filespec
--------K-2FE102-----------------------------
INT 2F - Phantom2 v1.1+ - UNINSTALL
  AX = E102h
Return: AX = status
      0001h removal successful
      0002h not installed as TSR
      FFFFh disabled but not removed
SeeAlso: AX=E100h
--------K-2FE103-----------------------------
INT 2F - Phantom2 v2.8 - SET ??? FLAG
  AX = E103h
Return: AX = 0001h
SeeAlso: AX=E100h
--------y-2FE200-----------------------------
INT 2F - SecureDevice - LOGIN TO DRIVE
  AX = E200h
  DL = drive number (0 = A:)
  DS:SI -> 104-byte key
Return: AL = status
      00h unable to determine key's validity
      01h key is valid
      FFh key is invalid
Program: SecureDevice is a copylefted device driver by Max Loewenthal and
    Arthur Helwig which turns one or more disk files into encrypted
    logical drives
SeeAlso: AX=E201h,AX=E203h,AX=E209h
--------y-2FE201-----------------------------
INT 2F - SecureDevice - GET INFORMATION
  AX = E201h
  DX = driver index (0000h = first loaded)
Return: AL = number of volumes handled by driver
  DL = drive number of first volume (00h = A:)
SeeAlso: AX=E200h,AX=E203h,AX=E209h
--------y-2FE203-----------------------------
INT 2F - SecureDevice - DESTROY PASSWORD (LOGOUT FROM DRIVE)
  AX = E203h
  DL = drive number (00h = A:) or FFh for all drives
```

```
Return: nothing
SeeAlso: AX=E200h,AX=E209h
--------y-2FE209DX0000----------------------
INT 2F - SecureDevice - INSTALLATION CHECK
  AX = E209h
  DX = 0000h
Return: AX = 1DEAh if installed
      DX = number of drivers installed
--------K-2FE300----------------------------
INT 2F - ANARKEY.COM - INSTALLATION CHECK
  AX = E300h
Return: AL = state
      00h not installed
      FEh if installed but suspended (v3.0+)
      FFh installed
Program: ANARKEY.COM is a commandline recall program by Steven Calwas
Range:  AH=C0h to AH=FFh, selected by commandline switch
SeeAlso: AX=E100h,AX=E301h,AX=E302h,AX=E303h,AX=E304h,AX=E305h,AX=E306h
SeeAlso: AX=E307h,INT 66"Newkey"
--------V-2FE300----------------------------
INT 2F - Blank - INSTALLATION CHECK
  AX = E300h
Return: AL = FFh if installed
      ES = resident code segment
Program: Blank is a shareware screen blanker by Yonah Schmeidler
Note: AH=E3h is the default, which may be reconfigured by the installation
    program in the registered version
SeeAlso: AH=93h,AX=C050h,INT 14/AX=AA01h
Index:  screen saver;Blank
--------K-2FE301----------------------------
INT 2F U - ANARKEY.COM v2+ - GET ???
  AX = E301h
Return: DX:BX -> ??? (see #03108,#03109)
SeeAlso: AX=E300h

Format of returned data structure for ANARKEY v2.0:
Offset  Size  Description (Table 03108)
 -7   7 BYTEs signature ('ANARKEY')
 00h  WORD  ??? (I see 0001h in v2.0)
 02h  WORD  ??? (I see 0001h in v2.0)
 04h  WORD  ??? (I see 0 in v2.0)
```

```
 06h  WORD  PSP segment of next program loaded

Format of returned data structure for ANARKEY v3+:
Offset  Size  Description (Table 03109)
 -1 BYTE  multiplex number
 00h  WORD  ??? (I see 0001h in v3.0-4.0)
 02h  WORD  ??? (I see 0001h in v3.0-4.0)
 04h  BYTE  ??? (I see 0 in v3.0-4.0)
 05h  WORD  PSP segment of next program loaded
--------K-2FE302-----------------------------
INT 2F U - ANARKEY.COM v3+ - ???
  AX = E302h
  BL = ???
Return: ???
SeeAlso: AX=E300h
--------K-2FE303-----------------------------
INT 2F U - ANARKEY.COM v3+ - ANARKMD API
  AX = E303h
  BL = function
       01h toggle insert mode
       02h display contents of history buffer
       03h write history buffer to file
    ES:DX -> file name
       04h clear history buffer
       05h undefine all aliases
       06h show aliases
       07h list programs using Unix switchar
       08h jump to bottom of history buffer
       09h (v4.0) add string to history buffer
    ES:DX -> ASCIZ string
       0Ah (v4.0) ???
    ES:DX -> ???
       0Bh (v4.0) copy string to edit buffer for use as next input line
    ES:DX -> ASCIZ string
       0Ch (v4.0) ???
       0Dh (v4.0) copy ??? to ???
       0Eh (v4.0) ???
       0Fh (v4.0) ???
       10h (v4.0) set ??? flag
       11h (v4.0) display error message about running in EMS under Windows
Return: ???
```

```
SeeAlso: AX=E300h
--------K-2FE304-----------------------------
INT 2F U - ANARKEY.COM v2+ - ???
  AX = E304h
  BL = ???
Return: ???
SeeAlso: AX=E300h
--------K-2FE305-----------------------------
INT 2F U - ANARKEY.COM v3+ - ENABLE/SUSPEND ANARKEY
  AX = E305h
  BL = new state
       01h suspended
       00h enabled
SeeAlso: AX=E300h
--------K-2FE306-----------------------------
INT 2F U - ANARKEY.COM v4.0 - GET ???
  AX = E306h
Return: AX = ???
SeeAlso: AX=E300h
--------K-2FE307-----------------------------
INT 2F U - ANARKEY.COM v4.0 - GET ???
  AX = E307h
Return: AX = ???
  BL = ???
SeeAlso: AX=E300h
--------K-2FE337-----------------------------
INT 2F - INT16.COM - INSTALLATION CHECK
  AX = E337h
Return: AX = 0013h if installed
Program: INT16.COM is an INT16 keyboard BIOS replacement for
    INT 16/AH=00h-02h,10h-12h (based on K3) written by Martin Gerdes
    and published in c't 05/1990.
SeeAlso: AX=ED58h
--------l-2FE44D-----------------------------
INT 2F - NDOS - API
  AX = E44Dh
Program: NDOS is a version of 4DOS licensed to Symantec for inclusion in the
    Norton Utilities
Note: as NDOS is a licensed version of 4DOS v3.03, the API is identical to
    that for 4DOS, except that AH=E4h instead of D4h and the installation
    check returns AX=44EEh instead of AX=44DDh
```

```
SeeAlso: AX=D44Dh,AX=E44Eh
--------l-2FE44E-----------------------------
INT 2F C - NDOS - AWAITING USER INPUT
  AX = E44Eh
  BX = condition
       0000h NDOS is ready to display prompt
       0001h NDOS has displayed the prompt, about to accept user input
Return: handler must preserve SI, DI, BP, SP, DS, ES, and SS
SeeAlso: AX=E44Dh
--------K-2FE44FBX0000-----------------------
INT 2F - NDOS v4.0+ - KSTACK.COM - INSTALLATION CHECK
  AX = E44Fh
  BX = 0000h
Return: AX = 44EEh if installed
Program: NDOS is a version of 4DOS licensed to Symantec for inclusion in the
    Norton Utilities
Note: this function is also supported by ANSIPLUS v3.01+ and K3PLUS v6.20+,
    which emulate the 4DOS and NDOS keystack
SeeAlso: AX=D44Fh/BX=0000h,AX=E44Fh/BX=0001h
--------K-2FE44FBX0001-----------------------
INT 2F - NDOS v4.0+ - KSTACK.COM - PLACE KEYSTROKES INTO KEYSTACK
  AX = E44Fh
  BX = 0001h
  CX = number of keystrokes (01h-FFh)
  DS:DX -> keystroke list (one word per keystroke)
Return: AX = status
       0000h successful
       nonzero failed
  BX,CX,DX destroyed
Notes:  the keystrokes are the exact values to return from subsequent calls to
    INT 16 with AH=00h,01h,10h, or 11h, with the following exceptions:
    0000h causes subfunctions 01h and 11h to indicate an empty
          keyboard buffer
    FFFFh is followed by a word indicating the number of clock
          ticks to delay before the next faked keystroke
  v4.00 KSTACK overwrites any unread keystrokes from the previous
    invocation, and does not range-check CX; it will overwrite memory
    following the resident portion if CX is greater than 100h.
  this function is also supported by ANSIPLUS v3.01+ and K3PLUS v6.20+,
    which emulate the 4DOS and NDOS keystack
SeeAlso: AX=E44Fh/BX=0000h,INT 16/AH=00h,INT 21/AX=4403h"DOS"
```

```
--------U-2FE600CL30-------------------------
INT 2F - Virtual 486 - INSTALLATION CHECK
  AX = E600h
  CL = 30h
  BX = C0DEh
Return: BX = DEC0h if installed
Program: Virtual 486 is a 80486 CPU emulator for an 80386 by Solar Designer
----------2FE700BX4158-----------------------
INT 2F - Multiplex - ??? - INSTALLATION CHECK???
  AX = E700h
  BX = 4158h ("AX")
  CX = 4953h ("IS")
  DX = 4845h ("HE")
Return: AL = FFh if installed
      BX = 4C4Fh ("LO") if ??? installed
      CX = 4F4Bh ("OK")
      DX = 4F55h ("OU")
    ES:DI -> ???
Range:  AH=C0h to AH=FFh, selected by scanning AH=E7-FFh, then AH=C0h-E6h
Note: called by QDPMI when its Real to Protected Mode Switch Entry Point
    is called
--------f-2FE77EBX0000----------------------
INT 2F - CTDEMN - INSTALLATION CHECK
  AX = E77Eh
  BX = 0000h
  CX = 4F4Dh ('OM')
  DX = 5453h ('TS')
Return: AX = 7EE7h if installed
      BX = resident code segment
      CX = 6F6Dh ('om')
      DX = 7473h ('ts')
Program: CTDEMN is a file daemon TSR by Simultan AG
--------E-2FED00----------------------------
INT 2F - Phar Lap DOS EXTENDERS - INSTALLATION CHECK
  AX = ED00h
  BL = DOS extender ID (see #03110)
Return: AL = status
      00h not installed
      FFh installed
    SI = 5048h ("PH")
    DI = 4152h ("AR")
```

```
    CH = major version number
    CL = minor version number
    DX = flags
        bit 0: running under DPMI
        bit 1: running under Phar Lap VMM
    if running under DPMI:
        BX = DPMI version (BH = major, BL = minor)
SeeAlso: AH=A1h,AX=F100h,AX=FBA1h

(Table 03110)
Values for Phar Lap DOS extender ID:
 01h  286dosx v1.3+ (Software Development Kit)
 02h  286dosx v1.3+ (Run-Time Kit)
 03h  386dosx v4.0+ (SDK)
 04h  386dosx v4.0+ (RTK)
--------E-2FED03-----------------------------
INT 2F R - Phar Lap 386/DOS-Extender v4.1 - GET EXTENDER ENTRY POINT
  AX = ED03h
  CX = real-mode code segment
  DX = real-mode data segment
Return: CF clear if successful
      CX = protected-mode code segment selector
      DX = protected-mode data segment selector
      ES:DI -> real-mode entry point for calling protected-mode functions
          (see INT 21/AX=250Dh)
  CF set on error
      AX = error code
    0008h unable to allocate LDT descriptors
--------E-2FED10BL05------------------------
INT 2F - Pharlap DOS Extender - ???
  AX = ED10h
  BL = 05h
  ES:SI -> ??? structure
Return: AX = ???
  SI = ???
  DI = ???
SeeAlso: AX=ED00h,AX=ED11h
--------E-2FED11BL05------------------------
INT 2F - Pharlap DOS Extender - FATAL EXIT TO REAL MODE ???
  AX = ED11h
  BL = 05h
```

```
  CX = ???
  DX = ???
  ES:SI -> ??? structure
  SS:SP = new stack ???
Return: ???
Note: called immediately prior to terminating program with INT 21/AX=4CFFh
SeeAlso: AX=ED00h,AX=ED10h
--------K-2FED58----------------------------
INT 2F U - K5.COM - INSTALLATION CHECK
  AX = ED58h
Return: AX = 000Dh if installed
      ES = resident code segment
      BX = ??? (9999h)
      ???
Program: K5 is an extended keyboard driver by Martin Gerdes, based on his K3
SeeAlso: AX=E337h,INT 16/AX=AF20h
--------E-2FED80----------------------------
INT 2F - Phar Lap 286|DOS Extender Lite v2.5 - ???
  AX = ED80h
  BL = DOS extender ID (see #03110)
  SI = 5048h ("PH")
  DI = 4152h ("AR")
  ???
Return: ???
--------O-2FEDC8BX0000----------------------
INT 2F - Novell DOS 7+ - SECURITY.BIN - INSTALLATION CHECK
  AX = EDC8h ('EDC' = Novell European Development Centre)
  BX = 0000h
  CX = 0000h
  DX = 1234h
Return: ???
Note: called by Novell DOS 7 LOCK command during installation

(Table 04107)
Call Novell DOS SECURITY.BIN API with:
  AX??? = function
      0000h get status
      0001h authenticate password hash
      0002h get / change encrypted master key password???
      0003h ??? Set encrypted master key password
      0004h ??? Get logged-in user ID
```

```
      0005h ??? Set logged-in user ID
      0006h Get device restriction mask
      0007h Set device restriction mask
      0008h Disable PNW DB access
      0009h Enable PNW DB access
      000Ah Get user name
      000Bh Set user name
      000Ch Change user password hash
      000Dh Get last function
      000Eh Get Encoded ASCII password
      000Fh Set Encoded ASCII password
      0010h Get user seed
      0011h Set user seed
      !!! more info to follow
--------y-2FEE00-----------------------------
INT 2F - GRIDLOC.EXE - INSTALLATION CHECK
  AX = EE00h
Return: AL = FFh if installed
Program: GRIDLOC is a PC security program by Intelligent Security Systems, Inc.
SeeAlso: INT 21/AH=40h"NB.SYS"
--------U-2FEE00-----------------------------
INT 2F - XVIEW - INSTALLATION CHECK
  AX = EE00h
Return: AX = 00FFh if installed
Program: XVIEW is a hypertext viewer by Flambeaux Software, Inc.
--------N-2FEE00-----------------------------
INT 2F - WEB v4.02 - INSTALLATION CHECK
  AX = EE00h
Return: AL = status
      00h not installed
      FFh installed
Program: WEB is an IPX-based peer-to-peer network by Webcorp.
SeeAlso: AH=EEh"WEB",AX=EEF0h
--------U-2FEE01-----------------------------
INT 2F - XVIEW - POP UP GIVING TOPIC SEARCH KEYWORD
  AX = EE01h
  DS:DX -> ASCIZ string containing case-insensitive keyword to look up
Return: AX = status (see #03111)
Note: the specified keyword should be a hyperlink in the _IndexPage of some
    database; the current database is searched first
SeeAlso: AX=EE00h"XVIEW",AX=EE02h,AX=EE03h,AX=EE04h,AX=EE06h
```

```
(Table 03111)
Values for XVIEW function status:
 0000h  successful
 00F1h  unknown subfunction
 00F2h  unable to pop up
--------U-2FEE02-----------------------------
INT 2F - XVIEW - POP UP GIVING A PAGE NUMBER
  AX = EE02h
  DX = physical page number or anchor page number (see #03112)
Return: AX = status (see #03111)
Note: physical page numbers are assigned by the hypertext compiler, and
    will change if a page is inserted in the middle
SeeAlso: AX=EE00h"XVIEW",AX=EE01h,AX=EE05h,AX=EE06h

(Table 03112)
Values for XVIEW anchor page number:
 FFEAh  _Credits
 FFECh  _SearchTopics
 FFEDh  _SearchText
 FFF0h  _ManualList
 FFF5h  _HelpOnHelp
 FFF8h  _HomePage
 FFF9h  _IndexPage
--------U-2FEE03-----------------------------
INT 2F - XVIEW - POP UP GIVING FILENAME AND SEARCH TOPIC OR PAGE NUMBER
  AX = EE03h
  DS:DX -> data packet (see #03113)
Return: AX = status (see #03111)
SeeAlso: AX=EE00h"XVIEW",AX=EE01h,AX=EE02h,AX=EE06h

Format of XVIEW data packet:
Offset  Size  Description (Table 03113)
 00h  DWORD -> ASCIZ database filespec (0000h:0000h for current database)
 04h  DWORD -> ASCIZ text to look up or 0000h:0000h
 08h  WORD  page number (0000h if keyword used)
 0Ah  6 BYTEs reserved
--------U-2FEE04-----------------------------
INT 2F - XVIEW - POP UP AND READ SCREEN FOR SEARCH TOPIC KEYWORD
  AX = EE04h
Return: AX = status (see #03111)
```

```
Note: equivalent to the action taken when the user presses the Alt-L hotkey
SeeAlso: AX=EE00h"XVIEW",AX=EE01h,AX=EE03h,AX=EE06h
--------U-2FEE05-----------------------------
INT 2F - XVIEW - POP UP TO MOST-RECENTLY VIEWED PAGE
  AX = EE05h
Return: AX = status (see #03111)
Note: equivalent to the action taken when the user presses the Alt-H hotkey
SeeAlso: AX=EE00h"XVIEW",AX=EE02h,AX=EE06h
--------U-2FEE06-----------------------------
INT 2F - XVIEW - WAIT FOR POP-DOWN AND GET EXIT CODE
  AX = EE06h
Return: AX = status (see also AX=EE01h)
      0001h specified filename is not an xText database
      0002h no databases found
      0003h bad data in file
      0004h memory shortage
      0005h unable to open the requested file
      0007h invalid page number for file
Note: although this call is not required, the exit code can alert the
    caller to problems; if the call is not made, the program should
    enforce a delay of about 1/2 second to allow the viewer to pop up,
    and should not get keyboard input or attempt disk accesses during
    the delay
SeeAlso: AX=EE00h"XVIEW",AX=EE01h,AX=EE02h,AX=EE03h,AX=EE04h,AX=EE05h
--------N-2FEE-----------------------------
INT 2F - WEB v4.02 - WEB MODULE INSTALLATION CHECK
  AH = EEh
  AL = module ID (see #03114)
Return: AX = 0000h if installed
      ES:DI -> far entry point for module-specific API calls
      (see #03115,#03116,#03117,#03118,#03119)
Program: WEB is an IPX-based peer-to-peer network by Webcorp.
SeeAlso: AX=EE00h"WEB"

(Table 03114)
Values for WEB module ID:
 10h  server module (SERVER.EXE)
 20h  client module (CLIENT.EXE)
 30h  mail module (MAIL.EXE)
 40h  spooler (PCSPOOL.EXE)
 50h  kernel module (KERNEL.EXE)
```

```
 60h  SAP module (KERNEL.EXE)
 70h  resident station manager (SM.EXE)
 90h  router module (ROUTER.EXE)

(Table 03115)
Call server module entry point with:
  BX = function
       0000h remove server module
    Return: AX = status (0000h if successful, else WEB error code)
       0001h create SYSINFO file
    Note: the SYSINFO file is used by the station manager when
        displaying info for a particular station
       0002h get server object table
    Return: CX = number of server objects
      ES:DI -> server object table
    Note: server objects include drives and devices that the
        server module controls
       0003h get server variables
    Return: ES:DI -> server variables

(Table 03116)
Call client module entry point with:
  BX = function
       0000h remove client module
    Return: AX = status (0000h if successful, else WEB error code)
       0001h decrement client-only flag
       0002h increment client-only flag
       0005h set device capture
    Note: decrements DeviceOutput flag, telling the spooler that
        it may trap device output again
       0006h clear device capture
    Note: increments DeviceOutput flag, telling the spooler that
        it should not trap device output (this is used
        internally by the spooler to prevent it from trapping
        its own output)
       0007h get client debug pointer
    Return: ES:DI -> client debug data structure (see #03120)
       0008h get root drive
    Return: AL = WEB startup drive
       0009h get maximum possible drive/device redirections
    Return: AL = maximum drive redirections
```

```
      CH = maximum LPTx redirections
      CL = maximum COMx redirections
      000Ah suspend client
    Return: AX = previous value of Suspend flag
      000Bh resume client
    Return: AX = previous value of Suspend flag
      000Ch get instance data
    CX = maximum number of structures in array
    ES:DI -> buffer for array of WIN_INSTANCE_DATA structures
        (see #03121)
    Return: CX = number of structures actually returned
    Note: used internally by WEB4WIN

(Table 03117)
Call mail module entry point with:
  BX = function
      0000h remove mail module
    Return: AX = status (0000h successful, else WEB error code)
      0001h set mail poll
    Note: schedules the WEB mail module
      0002h set mail notify
    Note: sets the Notify flag, which determines whether the
        user will be notified when mail is received
      0003h clear mail notify
    Note: clears the Notify flag, which determines whether the
        user will be notified when mail is received
      0004h check whether new mail has arrived
    Return: AL = new mail status
          00h no new mail since last call
          else new mail has arrived
    Note: also clears the new-mail flag after retrieving it
      0005h send notify
    ES:DI -> name of WEB user to be notified
      0006h get post office
    Return: ES:DI -> full network path of Post Office subdirectory

(Table 03118)
Call spooler entry point with:
  BX = function
      0000h remove PCSpool module
    Return: AX = status (0000h successful, else WEB error code)
```

```
      0001h set spooler poll
    Note: schedules the WEB spooler
      0002h check spooler changed
    Return: AX = 0000h
    Note: this call is a NOP in current versions of WEB

(Table 03119)
Call kernel entry point with:
  BX = function
      0000h remove kernel module
    Return: AX = status (0000h successful, else WEB error code)
      0001h set kernel ^S filter
    DL = new state (00h don't filter ^S, nonzero do filter)
      0002h get kernel data area
    Return: ES:DI -> kernel data area
      0003h display dialog box
    CL = dialog box type
        00h password
        01h E-Note received notification
        02h Novell login
        03h general notification
    DL = number of rows to display
    ES:SI -> array of far pointers to rows to be displayed
    ES:DI -> Pascal-style input buffer
    Return: AX = status (0000h successful, else error code)
      0004h kernel service events
      0005h get kernel's in-critical-section flag
    Return: ES:DI -> kernel InCriticalSection flag
      0006h schedule DOS event
    AL = directive
        00h do not ignore WEB ExtraBusy flag
        01h ignore ExtraBusy flag
        02h (WEB4WIN) check that current Windows VM is foregrnd VM
    ES:SI -> WEB AES Event Control Block (ECB) (see #03122)
    Notes:  the WEB Asynchronous Event Scheduler is similar to the
        one used by IPX; this call schedules a special ECB
        to be executed at a later time.  Unlike IPX ECBs,
        the timeout must be set explicitly by the caller
      this function also calls function 0004h
      0007h check busy
    AL = directive
```

```
        00h do not ignore WEB ExtraBusy flag
        01h ignore ExtraBusy flag
        02h (WEB4WIN) check that current Windows VM is foregrnd VM
    Return: AX = status (0000h not busy, else busy)
      0008h set keyboard intercept
    Note: currently a NOP which returns immediately
      0009h get keyboard intercept
    Note: currently a NOP which returns immediately
      000Ah get dialog flags
    Return: ES:DI -> kernel dialog flags (see #03123)
      000Bh get network path
    Return: ES:DI -> fully-qualified network path of file where
        the screen is stored on Dialog calls
      000Ch kernel alternate dialog
    CL = dialog box type
        00h password
        01h E-Note received notification
        02h Novell login
        03h general notification
    DL = number of rows to display
    ES:SI -> array of far pointers to rows to be displayed
    ES:DI -> Pascal-style input buffer
    Return: AX = status (0000h successful, else error code)
    Note: this function is identical to function 0003h except
        that it does not notify WEB4WIN of the impending
        dialog request
      000Dh get machine/operating system type
    Return: AX = machine/operating system type
          01h IBM PC, MS-DOS
          02h IBM PC, DOSV (Japanese)
          03h NEC PC-9800, JDOS (Japanese)
          04h IBM PC, Korean DBC DOS

Format of client debug data structure:
Offset  Size  Description (Table 03120)
 00h  WORD  total files
 02h  WORD  files free
 04h  WORD  no files
 06h  WORD  minimum files
 08h  WORD  total FCBs
 0Ah  WORD  total safe FCBs
```

```
 0Ch  WORD  FCBs in use
 0Eh  WORD  wrong FCB
 10h  WORD  compressed
 12h  WORD  retransmits

Format of WIN_INSTANCE_DATA structure:
Offset  Size  Description (Table 03121)
 00h  DWORD real-mode pointer to data to be instanced
 04h  WORD  size of data to be instanced

Format of WEB AES Event Control Block:
Offset  Size  Description (Table 03122)
 00h  DWORD link address
 04h  WORD  ESR address
 08h  BYTE  InUse flag
 09h  BYTE  completion code
 0Ah  3 BYTEs reserved
 0Dh  WORD  timeout
 0Fh  BYTE  IgnoreExtra flag
 10h  WORD  PSP
 12h  DWORD DTA
 16h  WORD  AX value for DOS critical information
 18h  WORD  BX value for DOS critical information
 1Ah  WORD  CX value for DOS critical information
 1Ch  WORD  DX value for DOS critical information

(Table 03123)
Values for kernel dialog flags:
 01h  dialog will timeout
 02h  display stars instead of entered keystrokes
--------N-2FEEF0-----------------------------
INT 2F - WEB v4.02 - WEB GENERAL NOTIFICATION
  AX = EEF0h
  BX = notification function ID (see #03124)
Return: varies by notification function
Program: WEB is an IPX-based peer-to-peer network by Webcorp.
Note: the notification functions are used internally by WEB modules to notify
    other modules and external programs of actions or event, and should
    never be called by an application
SeeAlso: AX=EE00h"WEB"
```

```
(Table 03124)
Values for WEB Notification Function ID:
 00h  node added
 01h  node deleted
 02h  dial attempt
 03h  dial failed
 04h  file close
 05h  close connection
 07h  check Windows mode
 20h  link up
 21h  link down
--------K-2FF000-----------------------------
INT 2F U - 4MAP - INSTALLATION CHECK
  AX = F000h
Return: AX = 00FFh
Program: 4MAP is a keybinding program for 4DOS (see AX=D44Dh) by Ho-Ping Tseng
Note: returns AX=00FFh for any value of AL not listed here
SeeAlso: AX=D44Dh,AX=F001h,AX=F002h
--------K-2FF001-----------------------------
INT 2F U - 4MAP - GET KEY MAPPINGS
  AX = F001h
Return: ES:BX -> key mappings
SeeAlso: AX=F000h
--------K-2FF002-----------------------------
INT 2F U - 4MAP - INSERT CHARACTER INTO ???
  AX = F002h
  BL = character to insert
Return: AX = status
      0000h successful
      0001h buffer full
SeeAlso: AX=F000h,AX=F003h
--------K-2FF003-----------------------------
INT 2F U - 4MAP - INSERT CHARACTER INTO ???
  AX = F003h
  BL = character to insert
Return: AX = status
      0000h successful
      0001h buffer full
Program: 4MAP is a keybinding program for 4DOS (see AX=D44Dh) by Ho-Ping Tseng
SeeAlso: AX=F000h,AX=F002h
--------m-2FF1-------------------------------
```

```
INT 2F U - MIN-MEM v2.11 - INSTALLATION CHECK
  AH = F1h
  AL <> F1h
Return: AL = F1h if installed
Program: MIN-MEM is a shareware TSR manager by Biologic which permits up to 24
    popup TSRs to be loaded but swapped out to disk, EMS, or XMS.  One
    TSR at a time is brought back into memory at the user's request.
--------E-2FF100-----------------------------
INT 2F - DOS EXTENDER INSTALLATION CHECK
  AX = F100h
Return: AL = FFh if DOS extender present
      SI = 444Fh ("DO")
      DI = 5358h ("SX")
Note: supported or soon to be supported by Phar Lap, Rational, Ergo, and IGC
SeeAlso: AH=A1h,AX=ED00h,AX=FBA1h/BX=0081h,INT 15/AX=BF02h
--------T-2FF1------------------------------
INT 2F U - RTKernel v4.0 - INSTALLATION CHECK
  AH = F1h
  AL = 00h
Return: AX = FFFFh if present
      CX = 00F1h
Program: RTKernel is a DOS preemptive multitasking library for C/Pascal by
    On Time Informatik GmbH
--------W-2FF200-----------------------------
INT 2F - WINX - INSTALLATION CHECK
  AX = F200h
Return: AX = 00FFh if installed
Program: WINX is a DOS/Windows utilities by Al Williams which can be used to
    launch Windows applications from a DOS Box; it was published in
    "DOS and Windows Protected Mode-Programming with DOS Extenders"
    (Addison-Wesley) and should not be confused with the Windows driver
    of the same name which is part of the DESQview/X package
--------W-2FF201-----------------------------
INT 2F - WINX - RETURN ADDRESS OF SERVER BUFFER
  AX = F201h
Return: AX = status
      FFFFh if WINX is busy processing a different request
      0000h if successful
    BX:CX = address of server buffer (see #03125)

Format of WINX server buffer:
```

```
Offset  Size  Description (Table 03125)
 00h  BYTE  command/status
    00h buffer available
    01h buffer contains result
    02h change directory
    03h execute program
    FFh terminate windows portion of WINX
 01h  ? BYTEs command (03h) or directory (02h)
  or
 01h  DWORD result (01h)
--------W-2FF202-----------------------------
INT 2F - WINX - SET SERVER'S WORKING DIRECTORY
  AX = F202h
  BX:CX -> directory
Return: AX = status
      FFFFh if WINX is busy processing a different request
      0000h if successful
SeeAlso: AX=F200h,AX=F203h
--------W-2FF203-----------------------------
INT 2F - WINX - EXECUTE COMMAND
  AX = F203h
  BX:CX -> command
Return: AX = status
      0000h if successful
      FFFFh if WINX is busy processing a different request
SeeAlso: AX=F200h,AX=F202h
--------G-2FF400-----------------------------
INT 2F - FINDIRQ.COM - INSTALLATION CHECK
  AX = F400h
Return: AL = 01h if installed
Program: FINDIRQ is a program by Rick Knoblaugh published in the 9/28/93 issue
    of PC Magazine; when run as a TSR it can determine which IRQs are
    used only when a device is active
SeeAlso: AX=F401h
--------N-2FF400-----------------------------
INT 2F - PowerLAN - INSTALLATION CHECK???
  AX = F400h
  ???
Return: ???
Note: this function is called by PowerLAN's NET.EXE just prior to calling
    AX=F401h (get version)
```

```
SeeAlso: AX=F401h"PowerLAN"
--------N-2FF401----------------------------
INT 2F - PowerLAN - GET VERSION
  AX = F401h
Return: ES:BX -> WORD containing 100*version (in decimal)
Program: PowerLAN is a networking product by Performance Technology
SeeAlso: AX=F400h"PowerLAN",AX=F483h"PowerLAN"
--------G-2FF401CX5121----------------------
INT 2F - FINDIRQ.COM - GET HOOKED INTERRUPTS
  AX = F401h
  CX = 5121h ('Q!')
Return: AX:DX -> hooked interrupt table (see #03126)
SeeAlso: AX=F400h

Format of FINDIRQ hooked interrupt table:
Offset  Size  Description (Table 03126)
 00h  BYTE  1Ch
 01h  DWORD FINDIRQ's INT 1C handler
 05h  DWORD original INT 1C handler
 09h  BYTE  28h
 0Ah  DWORD FINDIRQ's INT 28 handler
 0Eh  DWORD original INT 28 handler
 12h  BYTE  2Fh
 13h  DWORD FINDIRQ's INT 2F handler
 17h  DWORD original INT 2F handler
--------N-2FF483----------------------------
INT 2F - PowerLAN - ???
  AX = F483h
  ???
Return: ???
SeeAlso: AX=F400h"PowerLAN",AX=F401h"PowerLAN"
--------d-2FF700----------------------------
INT 2F - AUTOPARK.COM - INSTALLATION CHECK
  AX = F700h
Return: AL = state
      00h not installed
      FFh installed
Program: AUTOPARK.COM is a resident hard disk parker by Alan D. Jones
--------d-2FF701----------------------------
INT 2F - AUTOPARK.COM - SET PARKING DELAY
  AX = F701h
```

```
  BX:CX = 32-bit count of 55ms timer ticks
--------d-2FF800CX4455-----------------------
INT 2F U - SuperStor PRO 2XON.COM - INSTALLATION CHECK
  AX = F800h
  CX = 4455h ("DU")
  DL = 45h ("E")
Return: AL = FFh if installed
      ES:BX -> ASCII signature "Universal Data Exchange"
Program: SuperStor is a disk-compression program by Addstor.
Note: returns AX=0001h if AL is not 00h or 01h
SeeAlso: AX=1001h,AX=F801h
--------d-2FF801CX4455-----------------------
INT 2F U - SuperStor PRO 2XON.COM - UNINSTALL
  AX = F801h
  CX = 4455h ("DU")
  DL = 45h ("E")
  ES:BX = return address if successful
Return: at specified address if successfully removed from memory
  else
      AL = error code
      ???
Program: SuperStor is a disk-compression program by Addstor.
Note: returns AX=0001h if AL is not 00h or 01h
SeeAlso: AX=1001h,AX=F800h
----------2FFA00-----------------------------
INT 2F - Multiplex - ??? - INSTALLATION CHECK???
  AX = FA00h
  BX = ??? (0408h)
  CX = ??? (001Fh)
  DX = ??? (0102h)
  SI = ??? (5ACCh)
  DI = ??? (0632h)
Return: ???
Note: called by WinEmacs at startup
----------2FFA00-----------------------------
INT 2F - Multiplex - ??? - INSTALLATION CHECK???
  AX = FA00h
  BX = ??? (03FCh)
  CX = ??? (003Fh)
  DX = ??? (00FFh)
  SI = ??? (5AA6h)
```

```
  DI = ??? (0620h)
Return: ???
Note: called by Matlab at startup
--------*-2FFB------------------------------
INT 2F - Multiplex - RESERVED BY BORLAND INTERNATIONAL
  AH = FBh
SeeAlso: AX=FB42h/BX=0001h
--------f-2FFB------------------------------
INT 2F U - Conner Backup Exec AUTORES - API
  AH = FBh
  BL = function number (00h-07h)
  ???
Return: ???
Program: AUTORES is a resident program launcher for unattended backups
--------a-2FFB00----------------------------
INT 2F U - AutoBraille v1.1A - INSTALLATION CHECK
  AX = FB00h
Return: AX = 00FFh if installed
Program: AutoBraille is a shareware text-to-braille converter by KANSYS, Inc.
SeeAlso: INT 10/AX=3800h,INT 14/AX=F0F1h
--------U-2FFB00----------------------------
INT 2F U - Jot-It! v1.50 - INSTALLATION CHECK
  AX = FB00h
Return: AX = FFFFh if installed
      BX = version (BCD, BH=major, BL=minor)
SeeAlso: AX=FB03h"Jot-It",AX=FB01h"Jot-It"
--------a-2FFB01----------------------------
INT 2F U - AutoBraille v1.1A - ???
  AX = FB01h
  ???
Return: ???
--------U-2FFB01----------------------------
INT 2F U - Jot-It! v1.50 - GET USER NAME
  AX = FB01h
Return: DX:BX -> ASCIZ user name
SeeAlso: AX=FB02h"Jot-It"
--------a-2FFB02----------------------------
INT 2F U - AutoBraille v1.1A - ???
  AX = FB02h
Return: AH = ???
  AL = ???
```

```
--------U-2FFB02-----------------------------
INT 2F U - Jot-It! v1.50 - GET MESSAGE DIRECTORY
  AX = FB02h
Return: DX:BX -> ASCIZ name of directory in which messages are stored
SeeAlso: AX=FB01h"Jot-It"
--------a-2FFB03-----------------------------
INT 2F U - AutoBraille v1.1A - GET NEXT ???
  AX = FB03h
Return: AX = ???
--------U-2FFB03-----------------------------
INT 2F U - Jot-It! v1.50 - UNINSTALL
  AX = FB03h
Return: resident code removed from memory
Note: CAUTION: NO checks are performed to ensure that the interrupt vectors
    being unhooked (08h,09h,28h,2Fh) actually point at the Jot-It! code
SeeAlso: AX=FB00h"Jot-It"
--------a-2FFB-------------------------------
INT 2F U - AutoBraille v1.1A - SET ???
  AH = FBh
  AL = 04h-08h
Return: AX = 0000h
--------a-2FFB-------------------------------
INT 2F U - AutoBraille v1.1A - SET ???
  AH = FBh
  AL = 09h-0Fh (???, 0Eh = COM1, 0Fh = COM2)
Return: ???
Program: AutoBraille is a shareware text-to-braille converter by KANSYS, Inc.
--------a-2FFB-------------------------------
INT 2F U - AutoBraille v1.1A - SET ???
  AH = FBh
  AL = 10h-1Fh
  ???
Return: ???
--------a-2FFB20-----------------------------
INT 2F U - AutoBraille v1.1A - SET ??? FLAGS
  AX = FB20h
  BL = flags to set
SeeAlso: AX=FB21h"AutoBraille"
--------a-2FFB21-----------------------------
INT 2F U - AutoBraille v1.1A - CLEAR ??? FLAGS
  AX = FB21h
```

```
  BL = flags to clear
SeeAlso: AX=FB20h"AutoBraille"
--------a-2FFB22----------------------------
INT 2F U - AutoBraille v1.1A - SET ???
  AX = FB22h
  BL = ???
Program: AutoBraille is a shareware text-to-braille converter by KANSYS, Inc.
--------a-2FFB28----------------------------
INT 2F U - AutoBraille v1.1A - ???
  AX = FB28h
  BX = ???
  ???
Return: ???
SeeAlso: AX=FB29h"AutoBraille"
--------a-2FFB29----------------------------
INT 2F U - AutoBraille v1.1A - ???
  AX = FB29h
  BX = ???
  ???
Return: ???
SeeAlso: AX=FB28h"AutoBraille"
--------a-2FFB----------------------------
INT 2F U - AutoBraille v1.1A - SET ???
  AH = FBh
  AL = 2Bh-34h
  BX = ???
--------a-2FFB35----------------------------
INT 2F U - AutoBraille v1.1A - SET ???
  AX = FB35h
  BL = ???
--------a-2FFB36----------------------------
INT 2F U - AutoBraille v1.1A - SET ???
  AX = FB36h
  BL = ???
--------a-2FFB37----------------------------
INT 2F U - AutoBraille v1.1A - SET ???
  AX = FB37h
  BL = ???
Program: AutoBraille is a shareware text-to-braille converter by KANSYS, Inc.
--------E-2FFB42BX0001----------------------
INT 2F PU - Borland C++ DPMILOAD.EXE - INSTALLATION CHECK???
```

```
  AX = FB42h
  BX = 0001h
Return: AX = version number??? (AL=major, AH=minor)
  CX = next-selector increment
---BC2.0---
  ES:BX -> 80-byte buffer for ???
  DX = DPMI version
---BC3.0---
  BX = ??? (0000h)
  DX = ???
  ES:SI -> list of valid selectors ???
Notes:  The version of DPMILOAD distributed with BC++ v2.0 identifies itself
    as version 1.000, while the version distributed with BC++ 3.0
    identifies itself as version 1.0; the former is 10864 bytes, the
    latter 22180 bytes.  The BC2.0 version is a DPMI loader, while
    the BC3.0 version also adds a DPMI host and DOS extender
  the BC++ 2.0 version displays an error message if called with BX
    values other than 0001h-0008h
SeeAlso: AX=1687h,AX=FB42h/BX=1001h,AX=FB43h
--------E-2FFB42BX0002-----------------------
INT 2F PU - Borland C++ 2.0 DPMILOAD.EXE - ALLOCATE MEMORY
  AX = FB42h
  BX = 0002h
  CX = size in bytes
  DX = bit flags
      bit 2: set to allocate DOS memory, clear for DPMI memory
  SI = selector of descriptor to be modified to access allocated memory
  DI = selector of a second descriptor to be modified
Return: AX = ??? or 0000h on error
  CX:DX = linear base address of DPMI memory block
  SI:DI = handle for DPMI memory block or FFFFh:FFFFh
  ???
Note: two segment descriptors may be set if a code and an aliased data
    segment are required; if only one descriptor is needed, SI should
    equal DI on entry
BUG:  when allocating DOS memory, the code computes the linear address by
    multiplying the segment number by 4 rather than shifting by 4
SeeAlso: AX=FB42h/BX=0003h,AX=FB42h/BX=0008h,INT 31/AX=0501h
--------E-2FFB42BX0002-----------------------
INT 2F PU - Borland C++ 3.0 DPMILOAD.EXE - ALLOCATE MEMORY
  AX = FB42h
```

```
  BX = 0002h
  ES:SI -> memory block info (see #03127)
Return: ???
Note: The version of DPMILOAD distributed with BC++ v2.0 identifies itself
    as version 1.000, while the version distributed with BC++ 3.0
    identifies itself as version 1.0; the former is 10864 bytes, the
    latter 22180 bytes.
SeeAlso: AX=FB42h/BX=0003h,AX=FB42h/BX=0008h,INT 31/AX=0501h

Format of DPMILOAD memory block info:
Offset  Size  Description (Table 03127)
 00h  WORD  flags (see #03128)
 02h  DWORD block size in bytes
---DPMI memory block---
 06h  DWORD DPMI memory block handle
---DOS memory block---
 06h  WORD  real-mode segment of memory block
 08h  WORD  selector of memory block
---
 0Ah  DWORD linear address of memory
 0Eh  WORD  memory operation error code
    0008h no more free LDT descriptors
---if flags bit 0 clear---
 10h  WORD  code segment selector for memory block or 0000h or FFFFh
 12h  WORD  data alias selector for memory block or 0000h or FFFFh
---if flags bit 0 set---
 10h  WORD  data segment selector for memory block or 0000h or FFFFh
 12h  WORD  unused???

Bitfields for DPMILOAD memory block flags:
Bit(s)  Description (Table 03128)
 0  set if data segment rather than code segment
 1  information valid
 2  set if DOS memory block rather than DPMI memory block
 4  ???
 15 set if no LDT selectors for memory block???
SeeAlso: #03127
--------E-2FFB42BX0003----------------------
INT 2F PU - Borland C++ DPMILOAD.EXE - GET AVAILABLE MEMORY
  AX = FB42h
  BX = 0003h
```

```
Return: DX:AX = size of largest free block in paragraphs
    0000h:0000h on error (BC3.0 version only)
Note: AX and DX are destroyed on error, but no other error indicator is
    returned, under the BC++ 2.0 version of DPMILOAD
SeeAlso: AX=FB42h/BX=0002h
--------E-2FFB42BX0004----------------------
INT 2F PU - Borland C++ DPMILOAD.EXE - LOAD PROTECTED-MODE EXECUTABLE???
  AX = FB42h
  BX = 0004h
  DS:DX -> ASCIZ filename of protected-mode executable
Return: CX = selector of ??? or 0000h
---BC3.0---
  DX = status (0000h,FFF4h,others???) (see #03129)
Note: the filename may also be terminated by a CR rather than a NUL under the
    BC++ 3.0 version of DPMILOAD

(Table 03129)
Values for DPMILOAD function status:
 0000h  successful
 0001h  ??? failure
 0002h  invalid selector
 0004h  unknown error
 0008h  no more LDT descriptors available???
 FFDEh  unable to set descriptor
 FFDFh  unable to get segment base address
 FFE0h  ???
 FFF2h  invalid parameter value
 FFF4h  component of filename too long (name not in 8.3 format)
 FFF5h  pathname too long (>79 chars)
 FFF6h  ???
 FFF8h  ???
 FFF9h  index out of range
 FFFAh  ???
 FFFCh  invalid access to code segment???
 FFFEh  ???
 FFFFh  general error
--------E-2FFB42BX0005----------------------
INT 2F PU - Borland C++ DPMILOAD.EXE - GET ADDRESS OF ??? BY NAME
  AX = FB42h
  BX = 0005h
  CX = selector of DPMILOAD data (see #03130)
```

```
  DS:DX -> ASCIZ or CR-terminated name of ??? (case ignored)
Return: DX = status (see #03129)
      0000h successful
    AX:BX -> ??? FAR function (called with two words on top of stk)
      else
    BX destroyed
SeeAlso: AX=FB42h/BX=0006h,AX=FB42h/BX=000Eh

Format of DPMILOAD data:
Offset  Size  Description (Table 03130)
 00h 12 BYTEs ???
 0Ch  WORD  ??? bit flags
 0Eh 14 BYTEs ???
 1Ch  WORD  number of memory control records (see #03133)
 1Eh 25 BYTEs ???
 37h  BYTE  ??? bit flags
   bit 4: data valid???
 38h  4 BYTEs ???
 3Ch  WORD  ???
 3Eh 12 BYTEs ???
 46h  BYTE  ??? counter
 47h  BYTE  ???
 48h  BYTE  ???
 49h  BYTE  ???
 4Ah  WORD  ???
 4Ch  2 BYTEs ???
 4Eh  WORD  offset of array of 64-byte memory control records
 52h  WORD  offset of name list (see #03131)
 54h  4 BYTEs ???
 58h  WORD  offset of array of 6-byte objects (see #03132)
 5Ah  8 BYTEs ???
 62h  9 BYTEs ASCIZ name for ???
 6Bh  9 BYTEs ASCIZ name for ???
  ???

Format of name list entry [array]:
Offset  Size  Description (Table 03131)
 00h  BYTE  length of name (00h if end of array)
 01h  N BYTEs name
 N+1  WORD  1-based index into array of unknown 6-byte objects
```

```
Format of 6-byte objects:
Offset  Size  Description (Table 03132)
 00h  BYTE  ???
 01h  BYTE  ???
 02h  BYTE  ???
 03h  BYTE  1-based index of memory control record
 04h  WORD  ???

Format of memory control record:
Offset  Size  Description (Table 03133)
 00h 20 BYTEs memory block info (see #03127)
 14h  6 BYTEs ???
 1Ah  BYTE  ???
 1Bh  2 BYTEs ???
 1Dh  BYTE  ??? bit flags
 1Eh 14 BYTEs ???
 2Ch  DWORD pointer to ??? memory control record or 0000h:0000h
 30h  DWORD pointer to ??? memory control record or 0000h:0000h
 34h  DWORD pointer to next??? memory control record or 0000h:0000h
 38h  DWORD pointer to prev??? memory control record or 0000h:0000h
 3Ch  4 BYTEs ???
Note: the pointers at offsets 2Ch and 30h form a doubly-linked list, as do
    the pointers at offsets 34h and 38h
--------E-2FFB42BX0006-----------------------
INT 2F PU - Borland C++ DPMILOAD.EXE - GET ADDRESS OF ??? BY NUMBER
  AX = FB42h
  BX = 0006h
  CX = selector of DPMILOAD data (see #03130)
  DX = 1-based index into array of ??? 6-byte objects
Return: DX = status (see #03129)
      0000h successful
    AX:BX -> ??? FAR function (called with two words on top of stk)
      else
    BX destroyed
SeeAlso: AX=FB42h/BX=0005h,AX=FB42h/BX=000Eh
--------E-2FFB42BX0007-----------------------
INT 2F PU - Borland C++ 2.0 DPMILOAD.EXE - ???
  AX = FB42h
  BX = 0007h
  CX = selector of ???
Return: ???
```

```
Note: The version of DPMILOAD distributed with BC++ v2.0 identifies itself
    as version 1.000, while the version distributed with BC++ 3.0
    identifies itself as version 1.0; the former is 10864 bytes, the
    latter 22180 bytes.
--------E-2FFB42BX0007-----------------------
INT 2F PU - Borland C++ 3.0 DPMILOAD.EXE - ???
  AX = FB42h
  BX = 0007h
  CX = selector of DPMILOAD data (see #03130)
  ???
Return: DX = status (see #03129)
      0000h successful
    AX = ???
--------E-2FFB42BX0008-----------------------
INT 2F PU - Borland C++ 2.0 DPMILOAD.EXE - FREE MEMORY BLOCK
  AX = FB42h
  BX = 0008h
  CX = bit flags
      bit 2: set if DPMI memory, clear if DOS memory
  DX = selector of DOS memory block
  SI:DI = handle of DPMI memory block
Return: DX = 0000h on error, unchanged if succcessful
Note: The version of DPMILOAD distributed with BC++ v2.0 identifies itself
    as version 1.000, while the version distributed with BC++ 3.0
    identifies itself as version 1.0; the former is 10864 bytes, the
    latter 22180 bytes.
SeeAlso: AX=FB42h/BX=0002h
--------E-2FFB42BX0008-----------------------
INT 2F PU - Borland C++ 3.0 DPMILOAD.EXE - FREE MEMORY BLOCK
  AX = FB42h
  BX = 0008h
  ES:SI -> memory block info (see #03127)
Return: ???
SeeAlso: AX=FB42h/BX=0009h
--------E-2FFB42BX0009-----------------------
INT 2F PU - Borland C++ 3.0 DPMILOAD.EXE - RESIZE MEMORY BLOCK
  AX = FB42h
  BX = 0009h
  ES:SI -> memory block info (see #03127)
  ???
Return: ???
```

```
SeeAlso: AX=FB42h/BX=0008h"3.0"
--------E-2FFB42BX000A-----------------------
INT 2F RU - Borland C++ 3.0 DPMILOAD.EXE - INIT DPMI HOST AND SPAWN SUBSHELL
  AX = FB42h
  BX = 000Ah
  CX = 0001h
  DX = ???
  SI = ???
Return: after user exits subshell
Notes:  this call is used by DPMIRES; unlike most of the DPMILOAD calls, this
    function is not available in protected mode.
  the BC2.0 version of DPMILOAD is purely a DPMI loader, while the BC3.0
    version also adds a DPMI host and DOS extender.
SeeAlso: AX=FB42h/BX=0004h,AX=FB42h/BX=0015h
--------E-2FFB42BX000B-----------------------
INT 2F PU - Borland C++ 3.0 DPMILOAD.EXE - UNUSED
  AX = FB42h
  BX = 000Bh
--------E-2FFB42BX000C-----------------------
INT 2F PU - Borland C++ 3.0 DPMILOAD.EXE - FREE DESCRIPTORS FOR MEMORY BLOCK???
  AX = FB42h
  BX = 000Ch
  ES:SI -> memory block info ??? (see #03127)
Return: DX = status???
SeeAlso: AX=FB42h/BX=000Fh
--------E-2FFB42BX000D-----------------------
INT 2F PU - Borland C++ 3.0 DPMILOAD.EXE - SIMULATE REAL MODE INTERRUPT
  AX = FB42h
  BX = 000Dh
  CX = number of words to copy from protected-mode to real mode stack
  DL = interrupt number
  DH = flags
      bit 0: reset the interrupt controller and A20 line
  ES:DI -> real-mode call structure (see #03148 at INT 31/AX=0300h)
Return: CX = status
      0000h successful
      0001h failed
SeeAlso: INT 31/AX=0300h
--------E-2FFB42BX000E-----------------------
INT 2F PU - Borland C++ 3.0 DPMILOAD.EXE - GET ADDRESS OF ???
  AX = FB42h
```

```
  BX = 000Eh
  DS:DX -> ASCIZ or CR-terminated name of ???
Return: CX = selector of DPMILOAD data (see #03130) corresponding to name,
        0000h on error
SeeAlso: AX=FB42h/BX=0006h,AX=FB42h/BX=001Fh
--------E-2FFB42BX000F----------------------
INT 2F PU - Borland C++ 3.0 DPMILOAD.EXE - REALLOCATE LDT DESCRPS TO MEMBLK???
  AX = FB42h
  BX = 000Fh
  ES:SI -> memory block info (see #03127)
Return: ???
SeeAlso: AX=FB42h/BX=000Ch
--------E-2FFB42BX0010----------------------
INT 2F PU - Borland C++ 3.0 DPMILOAD.EXE - CONVERT SEGMENT TO SELECTOR
  AX = FB42h
  BX = 0010h
  DX = segment number
Return: CX = status (0000h,0008h) (see also AX=FB42h/BX=0004h)
      0000h successful
    DX = selector number for descriptor
      0008h failed
SeeAlso: AX=FB42h/BX=0023h
--------E-2FFB42BX0011----------------------
INT 2F PU - Borland C++ 3.0 DPMILOAD.EXE - ???
  AX = FB42h
  BX = 0011h
  CX = selector of DPMILOAD data (see #03130)
  ???
Return: DX = status (0000h,0002h,FFFEh) (see also #03129)
      0000h successful
    AX:BX -> ??? name
      FFFEh ??? error
--------E-2FFB42BX0012----------------------
INT 2F PU - Borland C++ 3.0 DPMILOAD.EXE - ???
  AX = FB42h
  BX = 0012h
  CX = selector for ???
Return: CX = selector for ???
--------E-2FFB42BX0013----------------------
INT 2F PU - Borland C++ 3.0 DPMILOAD.EXE - ???
  AX = FB42h
```

```
  BX = 0013h
  CX = selector of DPMILOAD data (see #03130)
  DX = 1-based index of ???
Return: CX = status (0000h,0002h,FFF9h) (see also #03129)
      0000h successful
    BX = ??? or 0000h
      FFF9h ??? error
--------E-2FFB42BX0014-----------------------
INT 2F RU - Borland C++ 3.0 DPMILOAD.EXE - INSTALLATION CHECK
  AX = FB42h
  BX = 0014h
  CX = 0001h
Return: BX = 0000h if installed
Note: unlike most of the DPMILOAD functions, this call is available only in
    real or V86 mode
SeeAlso: AX=FB42h/BX=0001h,AX=FB42h/BX=000Ah
--------E-2FFB42BX0015-----------------------
INT 2F RU - Borland C++ 3.0 DPMILOAD.EXE - UNINSTALL
  AX = FB42h
  BX = 0015h
  CX = 0001h
Return: ???
Note: unlike most of the DPMILOAD functions, this call is available only in
    real or V86 mode
SeeAlso: AX=FB42h/BX=000Ah
--------E-2FFB42BX0016-----------------------
INT 2F PU - Borland C++ 3.0 DPMILOAD.EXE - GET ???
  AX = FB42h
  BX = 0016h
  CX = selector of DPMILOAD data (see #03130)
Return: DX = status (see also AX=FB42h/BX=0004h)
      0000h successful
    CX = ???
--------E-2FFB42BX0017-----------------------
INT 2F PU - Borland C++ 3.0 DPMILOAD.EXE - ???
  AX = FB42h
  BX = 0017h
  CX = ???
  DX = ???
  ???
Return: DX = status (0000h,0001h) (see #03129)
```

```
--------E-2FFB42BX0018----------------------
INT 2F PU - Borland C++ 3.0 DPMILOAD.EXE - SET ???
  AX = FB42h
  BX = 0018h
  CX = ???
--------E-2FFB42BX0019----------------------
INT 2F PU - Borland C++ 3.0 DPMILOAD.EXE - ???
  AX = FB42h
  BX = 0019h
  CX = selector for ???
  ???
Return: DX = status (see also AX=FB42h/BX=0004h)
      0000h successful
    CX = selector for ???
--------E-2FFB42BX001A----------------------
INT 2F PU - Borland C++ 3.0 DPMILOAD.EXE - ???
  AX = FB42h
  BX = 001Ah
  CX = selector for ???
  ???
Return: DX = status (see also AX=FB42h/BX=0004h)
      0000h successful
      0004h failed
  CX:BX -> ???
--------E-2FFB42BX001B----------------------
INT 2F PU - Borland C++ 3.0 DPMILOAD.EXE - ???
  AX = FB42h
  BX = 001Bh
  CX = selector of DPMILOAD data (see #03130)
  DX = offset of ???
Return: DX = status (0000h,0002h) (see also #03129)
      0000h successful
    BX = selector for ???
    CX = selector for ???
--------E-2FFB42BX001C----------------------
INT 2F PU - Borland C++ 3.0 DPMILOAD.EXE - ???
  AX = FB42h
  BX = 001Ch
  ES = selector for DPMILOAD data (see #03130)
  CX = 1-based index of ???
  DX = 1-based index of ???
```

```
Return: DX = status (0000h,0002h,FFF9h) (see #03129)
--------E-2FFB42BX001D-----------------------
INT 2F PU - Borland C++ 3.0 DPMILOAD.EXE - GET ???
  AX = FB42h
  BX = 001Dh
Return: CX:DX = ???
--------E-2FFB42BX001E-----------------------
INT 2F PU - Borland C++ 3.0 DPMILOAD.EXE - ???
  AX = FB42h
  BX = 001Eh
  CX = ???
  ???
Return: DX = status (see also AX=FB42h/BX=0004h)
      0000h successful
      FFF7h ??? error
  CX:BX -> ???
--------E-2FFB42BX001F-----------------------
INT 2F PU - Borland C++ 3.0 DPMILOAD.EXE - GET ADDRESS OF ???
  AX = FB42h
  BX = 001Fh
  DS:DX -> 8-character name of ???
  ???
Return: CX = selector of DPMILOAD data (see #03130) for ???
    0000h on error
SeeAlso: AX=FB42h/BX=000Eh
--------E-2FFB42BX0020-----------------------
INT 2F PU - Borland C++ 3.0 DPMILOAD.EXE - NULL FUNCTION???
  AX = FB42h
  BX = 0020h
Return: DX = ??? (always 0000h)
--------E-2FFB42BX0021-----------------------
INT 2F PU - Borland C++ 3.0 DPMILOAD.EXE - GET PROCESSOR EXCEPTION HANDLER VECT
  AX = FB42h
  BX = 0021h
  CL = exception number (00h-1Fh)
Return: DX = status (see also AX=FB42h/BX=0004h)
      0000h successful
    AX:BX = selector:offset of handler
      FFF2h unable to get exception handler vector
SeeAlso: AX=FB42h/BX=0022h,AX=FB42h/BX=0024h,INT 31/AX=0202h
--------E-2FFB42BX0022-----------------------
```

```
INT 2F PU - Borland C++ 3.0 DPMILOAD.EXE - SET PROCESSOR EXCEPTION HANDLER VECT
  AX = FB42h
  BX = 0022h
  CL = exception number (00h-1Fh)
  SI:DX = selector:offset of new handler
Return: DX = status (0000h,0004h,FFF2h) (see #03129)
SeeAlso: AX=FB42h/BX=0021h,AX=FB42h/BX=0025h,INT 31/AX=0203h
--------E-2FFB42BX0023-----------------------
INT 2F PU - Borland C++ 3.0 DPMILOAD.EXE - CONVERT SELECTOR TO SEGMENT NUMBER
  AX = FB42h
  BX = 0023h
  CX = selector
Return: DX = status (see also AX=FB42h/BX=0004h)
      0000h successful
    CX = real-mode segment number
      FFF2h descriptor has invalid base address for real-mode segment
SeeAlso: AX=FB42h/BX=0010h
--------E-2FFB42BX0024-----------------------
INT 2F PU - Borland C++ 3.0 DPMILOAD.EXE - GET PROTECTED-MODE INTERRUPT VECTOR
  AX = FB42h
  BX = 0024h
  CL = interrupt number
Return: DX = status (0000h) (see also AX=FB42h/BX=0004h)
  AX:BX = selector:offset of handler
SeeAlso: AX=FB42h/BX=0025h,INT 31/AX=0204h
--------E-2FFB42BX0025-----------------------
INT 2F PU - Borland C++ 3.0 DPMILOAD.EXE - SET PROTECTED-MODE INTERRUPT VECTOR
  AX = FB42h
  BX = 0025h
  CL = interrupt number
  SI:DX = selector:offset of new handler
Return: DX = status (0000h,0004h,FFF2h) (see #03129)
SeeAlso: AX=FB42h/BX=0024h,INT 31/AX=0205h
--------E-2FFB42BX0026-----------------------
INT 2F PU - Borland C++ 3.0 DPMILOAD.EXE - ???
  AX = FB42h
  BX = 0026h
  CX = selector of DPMILOAD data (see #03130)
  DX = 1-based index of ???
  ???
Return: DX = status (0000h,0002h,FFF9h) (see #03129)
```

```
      0000h successful
    BX = offset of ??? within data structure
--------E-2FFB42BX0027-----------------------
INT 2F PU - Borland C++ 3.0 DPMILOAD.EXE - GET ???
  AX = FB42h
  BX = 0027h
  CX = selector of DPMILOAD data (see #03130)
  DX = offset of ???
Return: DX = status (see also AX=FB42h/BX=0004h)
      0000h successful
    BX = ???
--------E-2FFB42BX0080-----------------------
INT 2F U - ??? - CALLED BY Borland C++ 3.0 DPMILOAD.EXE
  AX = FB42h
  BX = 0080h
  ???
Return: AX = ???
  ???
--------E-2FFB42BX0081-----------------------
INT 2F U - ??? - CALLED BY Borland C++ 3.0 DPMILOAD.EXE
  AX = FB42h
  BX = 0081h
  ???
Return: AX = ???
  ???
--------E-2FFB42BX1001-----------------------
INT 2F U - Borland RTM.EXE 1.0 - INSTALLATION CHECK???
  AX = FB42h
  BX = 1001h
Return: BX = 0000h
SeeAlso: AX=FB42h/BX=0001h,AX=FB42h/BX=1002h,AX=FB42h/BX=1003h
--------E-2FFB42BX1002-----------------------
INT 2F U - Borland RTM.EXE 1.0 - EXECUTE COMPILED PROGRAM
  AX = FB42h
  BX = 1002h
  DX = 0014h ???
  ???
Return: AX = nonzero if successful
      AH = child program exit method??? (usually 4Ch)
      AL = child program exit code (Errorlevel)
  AX = 0000h on load error
```

```
      DX = error code (0000h-0027h) (see #03134)
SeeAlso: AX=FB42h/BX=1001h

(Table 03134)
Values for RTM.EXE error code:
 01h  out of memory
 02h  out of selectors
 03h  out of internal tables
 10h  internal error
 18h  internal error
 1Ah  internal error
 1Bh  internal error
 1Ch  internal error
 20h  invalid dynamic link
 21h  internal error
 22h  unable to open file
 23h  invalid .EXE format
 24h  wrong version
 25h  unable to initialize
 26h  DLL initialization error
 other  unrecognized error
--------E-2FFB42BX1003----------------------
INT 2F U - Borland RTM.EXE 1.0 - ???
  AX = FB42h
  BX = 1003h
  ???
Return: ???
SeeAlso: AX=FB42h/BX=1001h
--------E-2FFB43----------------------------
INT 2F PU - Borland C++ 3.0 DPMILOAD.EXE - NULL FUNCTION
  AX = FB43h
  BX = subfunction (at least 0000h-000Eh)
Notes:  this function is only present in protected mode; it does nothing but
    an immediate IRET
  DPMILOAD.EXE itself calls various subfunctions:
      subfunction 0004h is called with CX=selector of ???, DI=selector
      of DPMILOAD data
      subfunction 0008h is called with CX=selector of DPMILOAD data
SeeAlso: AX=FB42h/BX=0001h
--------G-2FFB43BX0100----------------------
INT 2F PU - Borland TDX - INSTALLATION CHECK
```

```
  AX = FB43h
  BX = 0100h
Return: BX = FB43h if loaded
Program: TDX is Borland's Turbo Debugger variant for DPMI programs
Note: Borland Pascal 7 DPMI programs use this call to check whether they
    should install their own stack and general protection exception
    handlers, or allow TDX to handle those exceptions
--------a-2FFB64-----------------------------
INT 2F U - AutoBraille v1.1A - GET ???
  AX = FB64h
Return: AX = ??? (0006h seen)
Program: AutoBraille is a shareware text-to-braille converter by KANSYS, Inc.
--------E-2FFBA1BX0081----------------------
INT 2F U - TKERNEL (Borland DOS extender) - INSTALLATION CHECK
  AX = FBA1h
  BX = 0081h
  ES:DI -> 16-byte buffer
Return: if installed, first four bytes of ES:DI buffer are "IABH"
Program: TKERNEL is a licensed version of AI Architects/Ergo's OS/x86.
Note: TKERNEL was present only in Borland C++ 2.0; with version 3.0, the DOS
    extender was moved into DPMILOAD.
SeeAlso: AH=A1h,AX=F100h,AX=FBA1h/BX=0082h,AX=FBA1h/BX=0084h,INT 15/AX=BF02h
SeeAlso: INT 21/AX=4403h"AI Architects"
--------E-2FFBA1BX0082----------------------
INT 2F U - TKERNEL (Borland DOS extender) - GET ENTRY POINT
  AX = FBA1h
  BX = 0082h
  ES:DI -> response buffer (see #03135)
Return: ES:DI buffer filled
SeeAlso: AX=FBA1h/BX=0081h,AX=FBA1h/BX=0084h

Format of TKERNEL response buffer:
Offset  Size  Description (Table 03135)
 00h  4 BYTEs signature "IABH"
 04h  DWORD pointer to FAR extender entry point (see #03136)

(Table 03136)
Call TKERNEL entry point with:
  AX = function number
       0000h initialize???
    STACK:  WORD  ???
```

```
Return: AX = status???
  STACK unchanged
  0001h get version???
Return: AX = 0200h for v2.0.34
  0002h get ???
Return: AX = ??? (011Eh or 0182h seen)
  0003h load protected-mode executable
STACK:  DWORD -> ASCIZ filename of executable
  DWORD ???
  DWORD -> program arguments (counted string plus CR)
  DWORD -> environment for protected-mode executable
    (terminated with two consecutive NULs)
  DWORD -> WORD buffer for ???
Return: AX = status???
  STACK unchanged
  0004h get descriptor
STACK:  WORD  selector for which to get descriptor
  WORD  segment number (when running in real mode)
  DWORD -> buffer for descriptor
Return: CF clear if successful
      buffer filled
  CF set on error
      AX destroyed???
  STACK unchanged
  0005h ???
STACK:  WORD  selector for ???
  WORD  subfunction number???
    0000h run previously-loaded program???
    0001h ??? (similar to 0000h)
    0002h
    0003h
    0005h ??? (similar to 0000h and 0001h)
Return: AX = status???
  STACK unchanged
  0006h ???
STACK:  WORD ???
  DWORD -> WORD (call) max iterations of ???
          (ret) remaining iterations
Return: AX = ???
  STACK unchanged
  0007h unused
```

```
Return: AX = 0001h
  0008h unused
Return: AX = 0001h
  0009h copy protected-mode memory into conventional memory
STACK:  WORD  selector for source segment
  WORD  segment of source if in real mode???
  DWORD offset of source
  WORD  number of bytes to copy
  DWORD -> low-memory destination
Return: AX = status
  STACK unchanged
  000Ah copy conventional memory into protected-mode memory
STACK:  WORD  selector for destination segment
  WORD  segment of destination if in real mode???
  DWORD offset of destination
  WORD  number of bytes to copy
  DWORD -> low-memory source
Return: AX = status
  STACK unchanged
  000Bh get ??? pointers
STACK:  WORD desired pointer
    0000h get ???
    0002h get protected-mode CR3
    0003h get 4K page table buffer pointer
    else Return: DX:AX = FFFFh:FFFFh
Return: DX:AX = requested pointer
  STACK unchanged
  000Ch set ??? pointers
STACK:  WORD desired pointer
    0000h set ???
    0002h set protected-mode CR3
    0003h set 4K page table buffer pointer
    else ignore
  DWORD new value for pointer
Return: STACK unchanged
  000Dh get ??? pointers
STACK:  WORD desired pointer
    0000h get ???
    0001h get ???
    0002h get ???
    0003h get ???
```

```
    0004h get ???
    0005h get ???
    0006h get ???
    0007h get ???
    else Return: DX:AX = FFFFh:FFFFh
Return: DX:AX = desired pointer
  STACK unchanged
  000Eh set ??? pointer
STACK:  WORD desired pointer
    0000h set ???
    0001h set ???
    0002h set ???
    0003h set ???
    0004h set ???
    0005h set ???
    0006h set ???
    0007h set ???
    else Return: DX:AX = FFFFh:FFFFh
Return: STACK unchanged
  000Fh get ???
Return: AX = ??? (seen 0008h)
  0010h get ???
Return: AX = ???
  0011h determine whether selector is valid
STACK:  WORD  possible selector
Return: AX = selector or 0000h if invalid
  STACK unchanged
  0012h get physical address
STACK:  WORD  selector for desired segment
  WORD  segment number if in real mode
  DWORD offset within segment
Return: DX:AX = 32-bit physical address or 00000000h on error
  BX destroyed
  STACK unchanged
  0013h ???
Note: normally jumps to code for function 0012h
  0014h copy protected-mode memory to conventional memory, with ???
STACK:  WORD  selector for source segment
  WORD  segment of source if in real mode???
  DWORD offset of source
  WORD  number of bytes to copy
```

```
    DWORD -> low-memory destination
Return: AX = status???
    STACK unchanged
    0015h copy conventional memory to protected-mode memory, with ???
STACK:  WORD  selector for destination segment
    WORD  segment of destination if in real mode???
    DWORD offset of destination
    WORD  number of bytes to copy
    DWORD -> low-memory source
Return: AX = status???
    STACK unchanged
    0016h set ??? pointer
STACK:  WORD  unused
    DWORD -> ??? or 0000h:0000h
Return: AX = 0000h
    STACK unchanged
    0017h allocate real-mode procedure???
STACK:  DWORD ASCIZ name of procedure
    DWORD ???
    DWORD address of subroutine to invoke
Return: AX = status
        0032h procedure by that name exists
        0033h no more real-mode procedures available
    DX destroyed
    STACK unchanged
    0018h unused
Return: AX = 0001h
    0019h get parameter block
Return: DX:AX -> parameter block (format unknown at this time,
        but 92h bytes)
        (preceded by signature "!!PARAM-BLOCK!!")
    001Ah get ???
Return: AX = ??? (0148h seen)
    001Bh free real-mode procedure???
STACK:  DWORD -> ASCIZ name of procedure
Return: ???
    STACK unchanged
    001Ch check whether packets from protected mode task pending
Return: AX = 0001h if packets pending, 0000h if not
    001Dh set ???
STACK:  DWORD ??? or 0000h:0000h
```

```
Return: AX,BX destroyed
  STACK unchanged
  001Eh ???
STACK:  WORD ??? (high byte ignored)
  DWORD -> data structure (below)
Return: AX,BX,CX,DX destroyed
  data structure updated
  STACK unchanged
Format of data structure:
Offset  Size  Description
 00h  2 BYTEs unused
 02h  WORD  ???
 04h  WORD  ???
 06h  WORD  ???
 08h  2 BYTEs unused
 0Ah  WORD  ???
 0Ch  WORD  (call) ???
    (ret) offset of this data structure (BUG?)
  001Fh set ???
STACK:  WORD ??? (set to 0001h if zero)
Return: AX destroyed
  STACK unchanged
  0020h ???
STACK:  DWORD -> ??? (8 bytes of data)
Return: AX = ???
  STACK unchanged
  0021h ???
STACK:  DWORD -> ??? (8 bytes of data)
  WORD  ???
  WORD  ???
Return: AX = ???
  STACK unchanged
  0022h ???
STACK:  DWORD -> ??? (8 bytes of data)
  DWORD -> 4-byte buffer for results
Return: AX = ???
  STACK unchanged
  0023h ???
STACK:  DWORD -> ??? (8 bytes of data)
Return: AX = ???
  STACK unchanged
```

```
      0024h set ???
    STACK:  WORD ???
    Return: AX destroyed
      STACK unchanged
      0025h get ???
    Return: AX = ??? (value set with func 0024h)
      0026h BUG: jumps to hyperspace due to fencepost error
      FFFFh set DOS memory management functions
    BX:SI -> FAR routine for allocating DOS memory
        (called with AH=48h,BX=number of paragraphs to alloc;
        returns CF clear, AX=segment of allocated memory, or
          CF set on error)
    CX:DI -> FAR routine for freeing DOS memory
        (called with AH=49h,ES=segment of block to free;
        returns CF set on error, AX=error code)
    Note: each of these pointers normally points at INT 21/RETF
      other Return: AX = 0001h
Note: BX may be destroyed by any of the API calls
--------E-2FFBA1BX0084-----------------------
INT 2F U - TKERNEL (Borland DOS extender) - UNINSTALL
  AX = FBA1h
  BX = 0084h
  ES:DI -> response buffer (see #03137)
Return: ES:DI buffer filled
SeeAlso: AX=FBA1h/BX=0081h,AX=FBA1h/BX=0084h

Format of TKERNEL response buffer:
Offset  Size  Description (Table 03137)
 00h  4 BYTEs signature "IABH"
 04h  WORD  success indicator
    0001h failed (INT 2F hooked by another program)
    unchanged if successful
 06h  WORD  segment of ???
 08h  WORD  segment of ??? memory block to free if nonzero
 0Ah  WORD  segment of ??? memory block to free if nonzero
--------s-2FFBFBES0000-----------------------
INT 2F U - SoundBlaster speech driver - INSTALLATION CHECK
  AX = FBFBh
  ES = 0000h
Return: ES nonzero if installed
      ES:BX -> entry point data structure (see #03138)
```

```
SeeAlso: INT 80/BX=0000h,INT F3"SoundBlaster"

Format of SoundBlaster entry point data structure:
Offset  Size  Description (Table 03138)
 00h  3 BYTEs signature "FB "
 03h  BYTE  driver major version number???
 04h  DWORD speech driver entry point (see #03139)
 08h 24 BYTEs ???
 20h  ? BYTEs data buffer for calling speech driver
      (can be 117 bytes or more)

(Table 03139)
Call SoundBlaster speech driver entry point with:
  AL = function
       07h speak a string
    data buffer (see #03138) contains:
      BYTE  length of string
          N BYTEs string to speak
--------K-2FFD12-----------------------------
INT 2F - KS/KEYSTKCT.EXE - INSTALLATION CHECK
  AX = FD12h
Return: AX = 0093h if installed
      ES = resident code segment
      ES:CX -> internal "putbuf" routine
Program: KS/KEYSTKCT.EXE is a key stacking utility (4DOS KEYSTACK.SYS
    look-alike) by Martin Gerdes, published in c't 11/1991, which can be
    loaded as a device driver or as a TSR. It does not emulate 4DOS
    KSTACK API
Note: the default buffer size is 128 keys
--------N-2FFE00BX4454-----------------------
INT 2F - PC-NFS ??? - INSTALLATION CHECK
  AX = FE00h
  BX = 4454h ("DT")
  CX = 4B52h ("KR")
  DX = 4E4Dh ("NM")
Return: AL = FFh if installed
     BX = 524Eh ("RM")
     CX = 4D44h ("MD")
     DX = 544Bh ("TK")
Note: DV/X 1.10 DVPCNFS.DVR searches AH=FEh,FFh,C0h-FDh for a valid response
SeeAlso: AX=FE08h
```

```
--------N-2FFE00BX4454-----------------------
INT 2F - PC-NFS ??? - INSTALLATION CHECK
  AX = FE00h
  BX = 4454h ("DT")
  CX = 4B52h ("KR")
  DX = 544Dh ("TM")
Return: AL = FFh if installed
     BX = 5254h ("RT")
     CX = 4D44h ("MD")
     DX = 544Bh ("TK")
Note: DV/X 1.10 DVPCNFS.DVR searches AH=FEh,FFh,C0h-FDh for a valid response
SeeAlso: AX=FE08h
--------U-2FFE00DI4E55----------------------
INT 2F U - NORTON UTILITIES 5.0+ TSRs - INSTALLATION CHECK/STATUS REPORT
  AX = FE00h
  DI = 4E55h ("NU")
  SI = TSR identifier (see #03140)
Return: SI = TSR reply
      lowercase version of SI on entry (i.e. SI ORed with 2020h)
      execpt SMARTCAN v8.0, which returns SI=6673h ('fs')
  AH = status
      00h installed but disabled internally
      01h installed and enabled
  AL = installed product
      00h NCACHE-x or DISKREET
      01h SPEEDRV / FILESAVE / EP / DISKMON v6+ installed
      02h NCACHE2 / SMARTCAN
      45h DISKMON v5 installed
  BX = length of *.INI file (DISKMON and FILESAVE/EP/SMARTCAN only)
       (see #03141,#03142)
  CX = segment of resident portion
      FFFFh if completely loaded high (NCACHE)
  DI may be destroyed
---FILESAVE/EP---
  DL = ??? (apparently always 00h)
---DISKMON---
  DX = ??? (apparently always 1AE6h [v5] / 1B86h [v6] / 1C26h [v7])
Notes:  the value returned in CX is incorrect for NCACHE 6.00
  all Norton Caches install as SMARTAAR drivers like SMARTDRV v3
  NCACHE2 and SPEEDRV both support the SMARTDRV v4+ installation check
  to detect Diskreet NDisk drives use CDS/DPB (see INT 21/AH=52h)
```

```
SeeAlso: AX=4A10h/BX=0000h,AX=FE01h,AX=FE02h,AX=FE03h,AX=FE04h,AX=FE05h
SeeAlso: INT 21/AX=4402h"SMARTDRV"

(Table 03140)
Values for Norton Utilities TSR identifier:
 4346h  ("CF") NCACHE-F (v5) / NCACHE (v6) / NCACHE2 (v7+) / SPEEDRV
 4353h  ("CS") NCACHE-S (v5 only)
 4443h  ("DC") DISKREET
 444Dh  ("DM") DISKMON
 4653h  ("FS") FILESAVE (v5) / EP (v6) / SMARTCAN (v7+)

Format of DISKMON.INI file:
Offset  Size  Description (Table 03141)
-6Ch 108 BYTEs  (in memory copy only)
    list of filenames which are always protected:
      IBMBIO.COM/IBMDOS.COM, IO.SYS/MSDOS.SYS, TBIOS.SYS/TDOS.SYS,
      MIO.SYS/IO.BIN, COMMAND.COM
 00h  BYTE  ??? always 01h
 01h  BYTE  disk light (00h off, 01h on)
 02h  BYTE  disk protection (00h off, 01h on)
 03h  BYTE  protected areas
    01h system area
    02h files
    03h system area and files
    04h entire disk
 04h  BYTE  floppy access (00h not allowed, 01h allowed)
 05h 27 BYTEs filename extension list (9 entries)
    (lowercase, blank padded or = 000000h)
 20h 240 BYTEs  filename list (20 entries)
    (lowercase, name and extension blank padded, with '.')
Note: CX:0508h -> copy in installed TSR (v5)
  CX:052Fh -> copy in installed TSR (v6)
  CX:04E0h -> copy in installed TSR (v7-v8)

Format of FILESAVE.INI / EP.INI / SMARTCAN.INI file:
Offset  Size  Description (Table 03142)
 00h 26 BITs  drive list (bit set: file protection on, cleared: off):
 00h  BYTE  drives  A: - H:
 01h  BYTE  drives  I: - P:
 02h  BYTE  drives  Q: - X:
 03h  BYTE  drives  Y: - Z:
```

```
 04h  BYTE  which files to protect
    00h all files
    01h all files with extension in list
    02h all files except those with extension in list
 05h 27 BYTEs filename extension list (9 entries, uppercase, ASCIZ)
 20h  BYTE  include files with archive bit clear (00h no, 01h yes)
 21h  WORD  number of days after which files are purged (0 = never)
 23h  WORD  max kilobytes of erased file space to hold (0 = all)
Note: CX:03D2h -> copy in installed TSR (v5)
  CX:03F5h -> copy in installed TSR (v6)
  CX:0434h -> copy in installed TSR (v7-v8)
--------U-2FFE00DX474F-----------------------
INT 2F - GO! v3.22+ - API
  AX = FE00h
  DX = 474Fh ('GO')
  SI = function number
      0063h (BCD for '?') installation check
      0078h (BCD for 'N') non-registered search (two levels only)
      0082h (BCD for 'R') reserved for registered version
      0083h (BCD for 'S') reserved for registered version
      0085h (BCD for 'U') uninstall
  BX:CX -> buffer (for search functions)
    buffer filled with search spec, i.e. "APL" to get first
      directory containing the substring APL, ":\APL" to find
      the first top-level directory beginning with the letters
      APL
Return: BX:CX buffer filled with result (search functions only)
    result is counted ASCIZ directory name, empty string if
    no matches (i.e. first byte is length of name, followed by
    name)
Program: GO! is a shareware directory locator TSR by Steve Ryckman
Note: the application-supplied buffer for the requests and results which
    BX:CX points at must lie outside the conventional (low-640K)
    memory, since the TSR swaps memory on pop-up; a common location is
    the last 96 bytes of the video memory or a UMB
--------U-2FFE01DI4E55----------------------
INT 2F U - NORTON UTILITIES 5.0+ TSRs - ENABLE
  AX = FE01h
  DI = 4E55h ("NU")
  SI = TSR identifier (see #03140)
Return: SI = TSR reply (lowercase version of entry SI, i.e. SI OR 2020h)
```

```
  AX = status
       0002h successful (DISKMON, FILESAVE, EP)
       FE00h successful (NCACHE-x, DISKREET)
Notes:  if the enable/disable calls are used on DISKMON or NCACHE-x, the status
    report generated by the programs still indicates the previous state,
    and DISKMON.INI is not updated
  apparently has no effect on DISKREET
SeeAlso: AX=FE00h,AX=FE02h
--------U-2FFE02DI4E55-----------------------
INT 2F U - NORTON UTILITIES 5.0+ TSRs - DISABLE
  AX = FE02h
  DI = 4E55h ("NU")
  SI = TSR identifier (see #03140)
Return: SI = TSR reply (lowercase version of entry SI, i.e. SI OR 2020h)
  AX = status
       0004h successful (DISKMON, FILESAVE)
       FE00h successful (NCACHE-x, DISKREET)
Notes:  (see also AX=FE01h)
  this function appears to be unsafe, as the cache buffers are not
    flushed
SeeAlso: AX=FE00h,AX=FE01h
--------U-2FFE03DI4E55-----------------------
INT 2F U - NORTON UTILITIES 5.0+ TSRs - FLUSH BUFFERS
  AX = FE03h
  DI = 4E55h ("NU")
  SI = TSR identifier (see #03140)
Return: SI = TSR reply (lowercase version of entry SI, i.e. SI OR 2020h)
  AX = status
       0006h successful???
Notes:  only supported by DISKMON, FILESAVE, and NCACHE-x
  useful for flushing NCACHE before rebooting
SeeAlso: AX=FE00h,AX=FE10h
--------U-2FFE04DI4E55-----------------------
INT 2F U - NORTON UTILITIES 5.0+ DISKMON, FILESAVE / EP - internal - ???
  AX = FE04h
  DI = 4E55h ("NU")
  SI = TSR identifier (see #03140)
Return: SI = TSR reply (lowercase version of entry SI, i.e. SI or 2020h)
  AX = status
       0008h successful???
SeeAlso: AX=FE00h
```

```
--------U-2FFE05DI4E55-----------------------
INT 2F U - NORTON UTILITIES 5.0+ DISKMON, FILESAVE / EP - internal - ???
  AX = FE05h
  DI = 4E55h ("NU")
  SI = TSR identifier (see #03140)
Return: SI = TSR reply (lowercase version of entry SI, i.e. SI or 2020h)
  AX = status
       000Ah successful???
Note: reportedly dangerous
SeeAlso: AX=FE00h
--------N-2FFE08----------------------------
INT 2F - PC-NFS ??? - GET ???
  AX = FE08h
Return: ES:BX -> ???
Notes:  DV/X 1.10 DVPCNFS.DVR searches AH=FEh,FFh,C0h-FDh for a valid response
  both the driver responding to AX=FE00h/DX=4E4Dh and the one responding
    to AX=FE00h/DX=544Dh support this function
SeeAlso: AX=FE00h/BX=4454h
--------U-2FFE10DI4E55-----------------------
INT 2F U - NORTON UTILITIES 6.0 NCACHE - REBOOT
  AX = FE10h
  DI = 4E55h ("NU")
  SI = TSR identifier (see #03140)
Return: SI = TSR reply (lowercase version of entry SI, i.e. SI or 2020h)
  AX = status
Note: probably used to flush NCACHE buffers and reboot when Ctrl-Alt-Del is
    detected
SeeAlso: AX=FE03h
--------S-2FFEEF----------------------------
INT 2F - RTS Control TSR - INSTALLATION CHECK
  AX = FEEFh
Return: AX = EFFEh if installed
      BX = port address
Program: RTS Control TSR is a utility by Michal Szokolo to lower the RTS
    signal on a COM port during disk accesses to avoid losing incoming
    data
--------N-2FFF00----------------------------
INT 2F - Topware Network Operating System - INSTALLATION CHECK
  AX = FF00h
Return: AL = status
      00h not installed, OK to install
```

```
      01h not installed, not OK to install
      FFh installed
SeeAlso: AX=FF01h,AX=FF02h,AX=FF10h,INT 21/AX=FF00h"Topware",INT 7A"Topware"
--------N-2FFF01----------------------------
INT 2F - Topware Network Operating System - GET VERSION
  AX = FF01h
Return: AX = version
SeeAlso: AX=FF00h,AX=FF02h
--------N-2FFF02----------------------------
INT 2F - TopWare Network OS v5.10+ - GET TopNet VERSION STRING
  AX = FF02h
Return: ES:BX -> version string
SeeAlso: AX=FF00h,AX=FF01h
--------N-2FFF10----------------------------
INT 2F - TopWare Network OS v5.10+ - TopTerm - INSTALLATION CHECK
  AX = FF10h
Return: AL = status (00h not installed, 01h installed)
SeeAlso: AX=FF00h,AX=FF11h,AX=FF12h,AX=FF13h
--------N-2FFF11----------------------------
INT 2F - TopWare Network OS v5.10+ - TopTerm - ENABLE KEYBOARD SERVICE
  AX = FF11h
Note: this function is only available on workstations, not on the server
SeeAlso: AX=FF10h,AX=FF12h
--------N-2FFF12----------------------------
INT 2F - TopWare Network OS v5.10+ - TopTerm - DISABLE KEYBOARD SERVICE
  AX = FF12h
Note: this function is only available on workstations, not on the server
SeeAlso: AX=FF10h,AX=FF11h
--------N-2FFF13----------------------------
INT 2F - TopWare Network OS v5.10+ - TopTerm - SET INSTALLATION FLAG
  AX = FF13h
  CL = new state (00h off, 01h on)
SeeAlso: AX=FF10h
--------N-2FFF14----------------------------
INT 2F - TopWare Network OS v5.10+ - START BACKGROUND RECEIVE VIDEO DATA
  AX = FF14h
Note: this function is only available on workstations, not on the server
SeeAlso: AX=FF10h,AX=FF15h
--------N-2FFF15----------------------------
INT 2F - TopWare Network OS v5.10+ - END BACKGROUND RECEIVE VIDEO DATA
  AX = FF15h
```

```
Note: this function is only available on workstations, not on the server
SeeAlso: AX=FF10h,AX=FF14h
--------N-2FFF16-----------------------------
INT 2F - TopWare Network OS v5.10+ - SET CONTROL NUMBER OF "SHOW" SCREEN
  AX = FF16h
  BL = which to set (00h TopShow, FFh TopTerm)
  CX = destination screen
       0000h all stations
       0000h-00FFh (TopTerm only) send to group CL
       8001h-80FEh send to station CL
SeeAlso: AX=FF18h
--------N-2FFF18-----------------------------
INT 2F - TopWare Network OS v5.10+ - SEND FULL SCREEN OF DATA FOR TopShow
  AX = FF18h
SeeAlso: AX=FF00h,AX=FF16h,AX=FF27h
--------N-2FFF23-----------------------------
INT 2F - TopWare Network OS v5.10+ - CLOSE SPOOL FILES AND START PRINTING
  AX = FF23h
SeeAlso: AX=FF00h
--------N-2FFF27-----------------------------
INT 2F - TopWare Network OS v5.10+ - GET "SHOW" TYPE
  AX = FF27h
Return: AL = type (00h complete version, 01h simple version)
  BL = "show" functions flag (00h disabled, 01h enabled)
SeeAlso: AX=FF16h,AX=FF18h
--------D-30--------------------------------
INT 30 - (NOT A VECTOR!) - DOS 1+ - FAR JMP instruction for CP/M-style calls
   the CALL 5 entry point does a FAR jump to here
Note: under DOS 2+, the instruction at PSP:0005 points two bytes too low in
    memory
SeeAlso: INT 21/AH=26h
--------V-30--------------------------------
INT 30 - QRIP/TSR - USED BY GRAPHICS LIBRARY
Program: QRIP/TSR is a shareware TSR by Shane Hathaway implementing the Remote
    Imaging Protocol (RIP, RIPscrip) used by several BBS systems to
    provide a graphical user interface
SeeAlso: INT 2F/AX=ACF0h
--------W-30--------------------------------
INT 30 P - MS Windows 3.1+ - PROTECTED-MODE CALLBACK
SeeAlso: INT 20"Windows"
--------D-31--------------------------------
```

```
INT 31 - overwritten by CP/M jump instruction in INT 30
--------v-31---------------------------------
INT 31 - VIRUS - "Vacsina" series - INSTALLATION CHECK (NOT A VECTOR!)
Note: if one of the Vacsina viruses is resident, the low byte of this
    interrupt still contains the last byte of the INT 30 CP/M JMP
    instruction, but the remaining three bytes are 7Fh 39h followed
    by the Vacsina version number
SeeAlso: INT 21/AX=FFFFh"VIRUS",INT 32"VIRUS"
--------E-310000-----------------------------
INT 31 P - DPMI 0.9+ - ALLOCATE LDT DESCRIPTORS
  AX = 0000h
  CX = number of descriptors to allocate
Return: CF clear if successful
      AX = base selector
  CF set on error
      AX = error code (DPMI 1.0+) (see #03143)
Notes:  DPMI is the DOS Protected-Mode Interface
  the base and limit of the returned descriptors will be 0, and the type
    will be "data"
  add the value returned by INT 31/AX=0003h to move to subsequent
    descriptors if multiple descriptors were allocated
  not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0001h,AX=000Dh,INT 21/AX=3501h

(Table 03143)
Values for DPMI 1.0 error code:
 0000h-7FFFh DOS error passed through by DPMI
 8001h  unsupported function
 8002h  object in wrong state for function
 8003h  system integrity would be endangered
 8004h  deadlock detected
 8005h  pending serialization request cancelled
 8010h  out of DPMI internal resources
 8011h  descriptor unavailable
 8012h  linear memory unavailable
 8013h  physical memory unavailable
 8014h  backing store unavailable
 8015h  callback unavailable
 8016h  handle unavailable
 8017h  maximum lock count exceeded
 8018h  shared memory already serialized exclusively by another
```

```
 8019h  shared memory already serialized shared by another client
 8021h  invalid value for numeric or flag parameter
 8022h  invalid segment selector
 8023h  invalid handle
 8024h  invalid callback
 8025h  invalid linear address
 8026h  request not supported by hardware
--------E-310001-----------------------------
INT 31 P - DPMI 0.9+ - FREE LDT DESCRIPTOR
  AX = 0001h
  BX = selector to free
Return: CF clear if successful
  CF set on error
      AX = error code (DPMI 1.0+) (8022h) (see #03143)
Notes:  only one descriptor is freed per call
  the program's initial CS, DS, and SS descriptors may be freed
  (DPMI 1.0+) any segment registers containing the freed selector are
    set to 0000h
  not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0000h,AX=000Ah,AX=000Dh,INT 21/AX=3502h
--------E-310002-----------------------------
INT 31 P - DPMI 0.9+ - SEGMENT TO DESCRIPTOR
  AX = 0002h
  BX = real mode segment
Return: CF clear if successful
      AX = selector corresponding to real mode segment (64K limit)
  CF set on error
      AX = error code (DPMI 1.0+) (8011h) (see #03143)
Notes:  multiple calls for the same real mode segment return the same selector
  the returned descriptor can never be modified or freed
  not supported by MS Windows 3.0 in Standard mode
--------E-310003-----------------------------
INT 31 P - DPMI 0.9+ - GET NEXT SELECTOR INCREMENT VALUE
  AX = 0003h
Return: CF clear
      AX = value to add to get next sequential selector
Notes:  the increment will be a power of two
  not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0000h
--------E-310004-----------------------------
INT 31 P - DPMI 0.9+ - LOCK SELECTOR
```

```
  AX = 0004h
  BX = selector to lock (prevent paging)
Return: ???
Note: although marked as reserved in versions 0.9 and 1.0 of the DPMI
    specification, this function is called by MS Windows TASKMAN,
    PROGMAN, and KERNEL
SeeAlso: AX=0005h,AX=0600h
--------E-310005----------------------------
INT 31 P - DPMI 0.9+ - UNLOCK SELECTOR
  AX = 0005h
  BX = selector to unlock (permit paging)
Return: ???
Note: although marked as reserved in versions 0.9 and 1.0 of the DPMI
    specification, this function is called by MS Windows TASKMAN,
    PROGMAN, and KERNEL
SeeAlso: AX=0004h,AX=0601h
--------E-310006----------------------------
INT 31 P - DPMI 0.9+ - GET SEGMENT BASE ADDRESS
  AX = 0006h
  BX = selector
Return: CF clear if successful
      CX:DX = linear base address of segment
  CF set on error
      AX = error code (DPMI 1.0+) (8022h) (see #03143)
Note: not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0007h,INT 21/AX=3504h
--------E-310007----------------------------
INT 31 P - DPMI 0.9+ - SET SEGMENT BASE ADDRESS
  AX = 0007h
  BX = selector
  CX:DX = linear base address
Return: CF clear if successful
  CF set on error
      AX = error code (DPMI 1.0+) (8022h,8025h) (see #03143)
Notes:  only modify descriptors allocated with INT 31/AX=0000h
  only the low 24 bits of the address will be used by 16-bit DPMI
    implementations even on a 386 or higher
  DPMI 1.0+ automatically reloads any segment registers containing the
    selector being modified
  not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0006h,AX=0008h,AX=0009h,AX=000Ch,INT 21/AX=3503h
```

```
SeeAlso: INT 21/AH=E9h"OS/286",INT 2C/AX=0002h
--------E-310008-----------------------------
INT 31 P - DPMI 0.9+ - SET SEGMENT LIMIT
  AX = 0008h
  BX = selector
  CX:DX = segment limit
Return: CF clear if successful
  CF set on error
      AX = error code (DPMI 1.0+) (8021h,8022h,8025h) (see #03143)
Notes:  CX must be zero for 16-bit DPMI implementations
  limits greater than 1MB must be page aligned (low 12 bits set)
  only modify descriptors allocated with INT 31/AX=0000h
  DPMI 1.0+ automatically reloads any segment registers containing the
    selector being modified
  not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0007h,AX=0009h,AX=000Ch,INT 21/AX=3505h,INT 21/AH=E9h"OS/286"
SeeAlso: INT 2C/AX=0003h,#00501 at INT 15/AH=89h
--------E-310009-----------------------------
INT 31 P - DPMI 0.9+ - SET DESCRIPTOR ACCESS RIGHTS
  AX = 0009h
  BX = selector
  CL = access rights/type byte (see #00502 at INT 15/AH=89h)
  CH = 80386 extended rights/type byte (see #00505 at INT 15/AH=89h)
      (32-bit DPMI implementations only)
Return: CF clear if successful
  CF set on error
      AX = error code (DPMI 1.0+) (8021h,8022h,8025h) (see #03143)
Notes:  if the Present bit is clear, CL bits 0-3 may have any value
  DPMI 1.0+ automatically reloads any segment registers containing the
    selector being modified
  not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0007h,AX=0008h,AX=000Ch,INT 21/AX=2514h,INT 2C/AX=0004h
SeeAlso: INT 2C/AX=0005h
--------E-31000A-----------------------------
INT 31 P - DPMI 0.9+ - CREATE ALIAS DESCRIPTOR
  AX = 000Ah
  BX = selector
Return: CF clear if successful
      AX = new data selector
  CF set on error
      AX = error code (DPMI 1.0+) (8011h,8022h) (see #03143)
```

```
Notes:  fails if selector in BX is not a code segment or is invalid
  use INT 31/AX=0001h to free new selector
  future changes to the original selector will not be reflected in the
    returned alias selector
  not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0001h
--------E-31000B-----------------------------
INT 31 P - DPMI 0.9+ - GET DESCRIPTOR
  AX = 000Bh
  BX = LDT selector
  ES:(E)DI -> 8-byte buffer for copy of descriptor
Return: CF clear if successful
      buffer filled
  CF set on error
      AX = error code (DPMI 1.0+) (8022h) (see #03143)
Notes:  16-bit programs use ES:DI as pointer, 32-bit must use ES:EDI
  not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=000Ch
--------E-31000C-----------------------------
INT 31 P - DPMI 0.9+ - SET DESCRIPTOR
  AX = 000Ch
  BX = LDT selector
  ES:(E)DI -> 8-byte buffer containing descriptor
Return: CF clear if successful
  CF set on error
      AX = error code (DPMI 1.0+) (8021h,8022h,8025h) (see #03143)
Notes:  16-bit programs use ES:DI as pointer, 32-bit must use ES:EDI
  only modify descriptors allocated with INT 31/AX=0000h
  DPMI 1.0+ automatically reloads any segment registers containing the
    selector being modified
  not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=000Bh
--------E-31000D-----------------------------
INT 31 P - DPMI 0.9+ - ALLOCATE SPECIFIC LDT DESCRIPTOR
  AX = 000Dh
  BX = LDT selector
Return: CF clear if successful
      descriptor allocated
  CF set on error
      AX = error code (DPMI 1.0+) (8011h,8022h) (see #03143)
Notes:  free descriptor with INT 31/AX=0001h
```

```
  the first 16 descriptors (04h-7Ch) are reserved for this function, but
    some may already be in use by other applications under DPMI 0.9;
    DPMI 1.0 guarantees 16 descriptors per client
  not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0000h,AX=0001h
--------E-31000E-----------------------------
INT 31 P - DPMI 1.0+ - GET MULTIPLE DESCRIPTORS
  AX = 000Eh
  CX = number of descriptors to copy
  ES:(E)DI -> descriptor buffer (see #03144)
Return: CF clear if successful
      descriptors copied
  CF set on error
      AX = error code (8022h) (see #03143)
      CX = number of descriptors successfully copied
Notes:  16-bit programs use ES:DI as pointer, 32-bit must use ES:EDI
  if the function fails, the first CX descriptors are valid; the
    remainder are not modified
SeeAlso: AX=000Bh,AX=000Fh

Format of DPMI descriptor buffer entry (one per descriptor to get):
Offset  Size  Description (Table 03144)
 00h  WORD  selector (set by client)
 02h  QWORD descriptor (set by host)
--------E-31000F-----------------------------
INT 31 P - DPMI 1.0+ - SET MULTIPLE DESCRIPTORS
  AX = 000Fh
  CX = number of descriptors to copy
  ES:(E)DI -> descriptor buffer (see #03145)
Return: CF clear if successful
      descriptors copied
  CF set on error
      AX = error code (8021h,8022h,8025h) (see #03143)
      CX = number of descriptors successfully copied
Notes:  16-bit programs use ES:DI as pointer, 32-bit must use ES:EDI
  if the function fails, the first CX descriptors are valid; the
    remainder are not modified
  DPMI 1.0+ automatically reloads any segment registers containing a
    selector being modified
SeeAlso: AX=000Ch,AX=000Eh
```

```
Format of DPMI descriptor buffer entry (one per descriptor to set):
Offset  Size  Description (Table 03145)
 00h  WORD  selector
 02h  QWORD descriptor
--------E-310100-----------------------------
INT 31 P - DPMI 0.9+ - ALLOCATE DOS MEMORY BLOCK
  AX = 0100h
  BX = number of paragraphs to allocate
Return: CF clear if successful
      AX = real mode segment of allocated block
      DX = first selector for allocated block
  CF set on error
      AX = DOS error code (07h,08h) (see #01680 at INT 21/AH=59h/BX=0000h)
    (DPMI 1.0+) DPMI error code (8011h) (see #03143)
      BX = size (in paragraphs) of largest available block
Notes:  multiple contiguous selectors are allocated for blocks of more than 64K
    if the caller is a 16-bit program
  never modify or deallocate returned descriptors
  not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0101h,AX=0501h
--------E-310101-----------------------------
INT 31 P - DPMI 0.9+ - FREE DOS MEMORY BLOCK
  AX = 0101h
  DX = selector of block
Return: CF set if successful
  CF set on error
      AX = DOS error code (07h,09h) (see #01680 at INT 21/AH=59h/BX=0000h)
Notes:  all descriptors allocated for the block are automatically freed
  DPMI 1.0+ automatically zeros any segment registers containing a
    selector freed by this function
  not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0100h,AX=0102h,AX=0502h
--------E-310102-----------------------------
INT 31 P - DPMI 0.9+ - RESIZE DOS MEMORY BLOCK
  AX = 0102h
  BX = new block size in paragraphs
  DX = selector of block
Return: CF clear if successful
  CF set on error
      AX = DOS error code (07h,08h,09h)
       (see #01680 at INT 21/AH=59h/BX=0000h)
```

```
    (DPMI 1.0+) DPMI error code (8011h,8022h) (see #03143)
      BX = maximum block size (in paragraphs) possible
Notes:  increasing the size of a block past a 64K boundary will fail if the
    next descriptor in the LDT is already in use
  shrinking a block past a 64K boundary will cause some selectors to be
    freed; DPMI 1.0+ automatically zeros any segment registers containing
    a selector freed by this function
  not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0100h
--------E-310200----------------------------
INT 31 P - DPMI 0.9+ - GET REAL MODE INTERRUPT VECTOR
  AX = 0200h
  BL = interrupt number
Return: CF clear
  CX:DX = segment:offset of real mode interrupt handler
Note: the DPMI implementation is required to support all 256 vectors
SeeAlso: AX=0201h,AX=0204h,INT 21/AX=2503h
--------E-310201----------------------------
INT 31 P - DPMI 0.9+ - SET REAL MODE INTERRUPT VECTOR
  AX = 0201h
  BL = interrupt number
  CX:DX = segment:offset of real mode handler
Return: CF clear
Note: all memory that may be touched by a hardware interrupt handler must be
    locked down with INT 31/AX=0600h
SeeAlso: AX=0200h,AX=0205h,AX=0600h,INT 21/AX=2505h
--------E-310202----------------------------
INT 31 P - DPMI 0.9+ - GET PROCESSOR EXCEPTION HANDLER VECTOR
  AX = 0202h
  BL = exception number (00h-1Fh)
Return: CF clear if successful
      CX:(E)DX = selector:offset of handler
  CF set on error
      AX = error code (DPMI 1.0+) (8021h) (see #03143)
Notes:  16-bit programs receive the pointer in CX:DX, 32-bit programs in CX:EDX
  DPMI 1.0+ supports this function only for backward compatibility; use
     AX=0210h or AX=0211h instead
  not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0203h,AX=0210h,AX=0211h,INT 2F/AX=FB42h/BX=0021h
--------E-310203----------------------------
INT 31 P - DPMI 0.9+ - SET PROCESSOR EXCEPTION HANDLER VECTOR
```

```
  AX = 0203h
  BL = exception number (00h-1Fh)
  CX:(E)DX = selector:offset of handler
Return: CF clear if successful
  CF set on error
      AX = error code (DPMI 1.0+) (8021h,8022h) (see #03143)
Notes:  32-bit programs must supply an offset in EDX and use a 32-bit interrupt
    stack frame on chaining to the next exception handler
  the handler should return using a FAR return
  all fault stack frames contain an error code, but it is only valid for
    exceptions 08h and 0Ah-0Eh
  handlers will only be called if the exception occurs in protected mode,
    and the DPMI host does not transparently handle the exception
  the handler may change certain values on the stack frame
    (see #03146,#03147)
  DPMI 1.0+ supports this function only for backward compatibility; use
     AX=0212h or AX=0213h instead
  not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0202h,AX=0212h,AX=0213h,INT 2F/AX=FB42h/BX=0022h

Format of stack frame for 16-bit programs: (offset from SS:SP)
Offset  Size  Description (Table 03146)
 00h  DWORD return CS:IP (do not change)
 04h  WORD  error code
 06h  DWORD CS:IP of exception
 0Ah  WORD  flags
 0Ch  DWORD SS:SP

Format of stack frame for 32-bit programs: (offset from SS:ESP)
Offset  Size  Description (Table 03147)
 00h  DWORD return EIP (do not change)
 04h  WORD  return CS selector (do not change)
 06h  WORD  reserved (do not change)
 08h  DWORD error code
 0Ch  DWORD EIP of exception
 10h  WORD  CS selector of exception
 12h  WORD  reserved (do not change)
 14h  DWORD EFLAGS
 18h  DWORD ESP
 1Ch  WORD  SS
 1Eh  WORD  reserved (do not change)
```

```
--------E-310204-----------------------------
INT 31 P - DPMI 0.9+ - GET PROTECTED MODE INTERRUPT VECTOR
  AX = 0204h
  BL = interrupt number
Return: CF clear
  CX:(E)DX = selector:offset of handler
Notes:  16-bit programs use CX:DX, 32-bit programs use CX:EDX
  DPMI implementations are required to support all 256 vectors
  not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0200h,AX=0205h,INT 21/AX=2502h,INT 2C/AX=0006h
SeeAlso: INT 2F/AX=FB42h/BX=0024h
--------E-310205-----------------------------
INT 31 P - DPMI 0.9+ - SET PROTECTED MODE INTERRUPT VECTOR
  AX = 0205h
  BL = interrupt number
  CX:(E)DX = selector:offset of handler
Return: CF clear if successful
  CF set on error
      AX = error code (DPMI 1.0+) (8022h) (see #03143)
Notes:  16-bit programs use CX:DX, 32-bit programs use CX:EDX
  32-bit programs must use a 32-bit interrupt stack frame when chaining
    to the next handler
  DPMI implementations are required to support all 256 vectors
  hardware interrupts are reflected to the virtual machine's primary
    client, software interrupts to the current client
  not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0201h,AX=0204h,INT 21/AX=2504h,INT 2C/AX=0007h
SeeAlso: INT 2F/AX=FB42h/BX=0025h
--------E-310210-----------------------------
INT 31 P - DPMI 1.0+ - GET PROTECTED MODE EXTENDED PROCESSOR EXCEPTION HANDLER
  AX = 0210h
  BL = exception number (00h-1Fh)
Return: CF clear if successful
      CX:(E)DX = selector:offset of exception handler
  CF set on error
      AX = error code (8021h) (see #03143)
Note: DPMI host reflects exception to current client's handler
SeeAlso: AX=0202h,AX=0211h,AX=0212h
--------E-310211-----------------------------
INT 31 P - DPMI 1.0+ - GET REAL MODE EXTENDED PROCESSOR EXCEPTION HANDLER
  AX = 0211h
```

```
  BL = exception number (00h-1Fh)
Return: CF clear if successful
      CX:(E)DX = selector:offset of exception handler
  CF set on error
      AX = error code (8021h) (see #03143)
Notes:  returns address of protected-mode handler for real-mode exception
  DPMI host performs a switch to protected mode, reflects the exception
    to the virtual machine's primary client, and returns to real mode
    on the handler's completion
SeeAlso: AX=0202h,AX=0210h,AX=0213h
--------E-310212-----------------------------
INT 31 P - DPMI 1.0+ - SET PROTECTED MODE EXTENDED PROCESSOR EXCEPTION HANDLER
  AX = 0212h
  BL = exception or fault number (00h-1Fh)
  CX:(E)DX = exception handler selector:offset
Return: CF clear if successful
  CF set on error
      AX = error code (8021h,8022h) (see #03143)
Note: DPMI host sends exception to current client's handler
SeeAlso: AX=0203h,AX=0210h,AX=0213h
--------E-310213-----------------------------
INT 31 P - DPMI 1.0+ - SET REAL MODE EXTENDED PROCESSOR EXCEPTION HANDLER
  AX = 0213h
  BL = exception or fault number (00h-1Fh)
  CX:(E)DX = exception handler selector:offset
Return: CF clear if successful
  CF set on error
      AX = error code (8021h,8022h) (see #03143)
Notes:  specifies address of protected-mode handler for real-mode exception
  DPMI host performs a switch to protected mode, reflects the exception
    to the virtual machine's primary client, and returns to real mode
    on the handler's completion
SeeAlso: AX=0203h,AX=0211h,AX=0212h
--------E-310300-----------------------------
INT 31 P - DPMI 0.9+ - SIMULATE REAL MODE INTERRUPT
  AX = 0300h
  BL = interrupt number
  BH = flags
      bit 0: reset the interrupt controller and A20 line (DPMI 0.9)
        reserved, must be 0 (DPMI 1.0+)
      others: reserved, must be 0
```

```
  CX = number of words to copy from protected mode to real mode stack
  ES:(E)DI = selector:offset of real mode call structure (see #03148)
Return: CF clear if successful
      real mode call structure modified (all fields except SS:SP, CS:IP
        filled with return values from real mode interrupt)
  CF set on error
      AX = error code (DPMI 1.0+) (8012h,8013h,8014h,8021h)(see #03143)
  protected mode stack unchanged
Notes:  16-bit programs use ES:DI as pointer, 32-bit programs use ES:EDI
  CS:IP in the real mode call structure is ignored for this call,
    instead, the indicated interrupt vector is used for the address
  the flags in the call structure are pushed on the real mode stack to
    form an interrupt stack frame, and the trace and interrupt flags are
    clear on entry to the handler
  DPMI will provide a small (30 words) real mode stack if SS:SP is zero
  the real mode handler must return with the stack in the same state as
    it was on being called
SeeAlso: AX=0302h,AX=FF01h,INT 21/AX=2511h,INT 21/AH=E3h"OS/286"
SeeAlso: INT 2C/AX=0026h,INT 2F/AX=FB42h/BX=000Dh

Format of DPMI real mode call structure:
Offset  Size  Description (Table 03148)
 00h  DWORD EDI
 04h  DWORD ESI
 08h  DWORD EBP
 0Ch  DWORD reserved (00h)
 10h  DWORD EBX
 14h  DWORD EDX
 18h  DWORD ECX
 1Ch  DWORD EAX
 20h  WORD  flags
 22h  WORD  ES
 24h  WORD  DS
 26h  WORD  FS
 28h  WORD  GS
 2Ah  WORD  IP
 2Ch  WORD  CS
 2Eh  WORD  SP
 30h  WORD  SS
--------E-310301-----------------------------
INT 31 P - DPMI 0.9+ - CALL REAL MODE PROCEDURE WITH FAR RETURN FRAME
```

```
  AX = 0301h
  BH = flags
       bit 0: reset the interrupt controller and A20 line (DPMI 0.9)
         reserved, must be 0 (DPMI 1.0+)
       others: reserved must be 0
  CX = number of words to copy from protected mode to real mode stack
  ES:(E)DI = selector:offset of real mode call structure
       (see #03148 at INT 31/AX=0300h)
Return: CF clear if successful
       real mode call structure modified (all fields except SS:SP, CS:IP
         filled with return values from real mode interrupt)
  CF set on error
       AX = error code (DPMI 1.0+) (8012h,8013h,8014h,8021h)(see #03143)
  protected mode stack unchanged
Notes:  16-bit programs use ES:DI as pointer, 32-bit programs use ES:EDI
  the real mode procedure must exit with a FAR return
  DPMI will provide a small (30 words) real mode stack if SS:SP is zero
  the real mode handler must return with the stack in the same state as
    it was on being called
SeeAlso: AX=0300h,AX=0302h,AX=FF02h,INT 21/AX=250Eh,INT 21/AH=E1h"OS/286"
SeeAlso: INT 2C/AX=0025h
--------E-310302----------------------------
INT 31 P - DPMI 0.9+ - CALL REAL MODE PROCEDURE WITH IRET FRAME
  AX = 0302h
  BH = flags
       bit 0: reset the interrupt controller and A20 line (DPMI 0.9)
         reserved, must be 0 (DPMI 1.0+)
       others: reserved, must be 0
  CX = number of words to copy from protected mode to real mode stack
  ES:(E)DI = selector:offset of real mode call structure
       (see #03148 at INT 31/AX=0300h)
Return: CF clear if successful
       real mode call structure modified (all fields except SS:SP, CS:IP
         filled with return values from real mode interrupt)
  CF set on error
       AX = error code (DPMI 1.0+) (8012h,8013h,8014h,8021h)(see #03143)
  protected mode stack unchanged
Notes:  16-bit programs use ES:DI as pointer, 32-bit programs use ES:EDI
  the flags in the call structure are pushed on the real mode stack to
    form an interrupt stack frame, and the trace and interrupt flags are
    clear on entry to the handler
```

```
  the real mode procedure must exit with an IRET
  DPMI will provide a small (30 words) real mode stack if SS:SP is zero
  the real mode handler must return with the stack in the same state as
    it was on being called
SeeAlso: AX=0300h
--------E-310303-----------------------------
INT 31 P - DPMI 0.9+ - ALLOCATE REAL MODE CALLBACK ADDRESS
  AX = 0303h
  DS:(E)SI = selector:offset of procedure to call
  ES:(E)DI = selector:offset of real mode call structure (see #03148)
Return: CF clear if successful
      CX:DX = segment:offset of real mode call address (see #03149)
  CF set on error
      AX = error code (DPMI 1.0+) (8015h) (see #03143)
Notes:  the real mode call structure is static, causing reentrancy problems;
    its contents are only valid at the time of a callback
  the called procedure must modify the real mode CS:IP before returning
  values are returned to real mode by modifying the real mode call struc
  DPMI hosts must provide at least 16 callbacks per client
  the limited DPMI host built into Phar Lap's 286|DOS-Extender v2.5 does
    not support this function
BUG:  Windows NT 4.0 either ignores or clears the high 16 bits of EDI,
    causing an illegal instruction error if the real mode call
    structure's offset in ES is greater than 64K
SeeAlso: AX=0304h,AX=0C00h

(Table 03149)
Values DPMI real-mode callback procedure is called with:
  DS:(E)SI = selector:offset of real mode SS:SP
  ES:(E)DI = selector:offset of real mode call structure
  SS:(E)SP = locked protected mode API stack
  interrupts disabled
Return: (with IRET)
  ES:(E)DI = selector:offset of real mode call structure to restore
--------E-310304-----------------------------
INT 31 P - DPMI 0.9+ - FREE REAL MODE CALLBACK ADDRESS
  AX = 0304h
  CX:DX = real mode callback address
Return: CF clear if successful
  CF set on error
      AX = error code (DPMI 1.0+) (8024h) (see #03143)
```

```
Note: the limited DPMI host built into Phar Lap's 286|DOS-Extender v2.5 does
    not support this function
SeeAlso: AX=0303h
--------E-310305-----------------------------
INT 31 P - DPMI 0.9+ - GET STATE SAVE/RESTORE ADDRESSES
  AX = 0305h
Return: CF clear
  AX = size in bytes of state buffer
  BX:CX = real mode address of procedure to save/restore state
  SI:(E)DI = protected mode procedure to save/restore state (see #03150)
Notes:  the buffer size will be zero if it is not necessary to preserve state
  16-bit programs should call SI:DI, 32-bit programs should call SI:EDI
  this function is only needed if using the raw mode switch service
SeeAlso: AX=0306h

(Table 03150)
Call DPMI state-save procedures with:
  AL = direction
       00h save state
       01h restore state
  ES:(E)DI -> state buffer
Return: all registers preserved
--------E-310306-----------------------------
INT 31 P - DPMI 0.9+ - GET RAW MODE SWITCH ADDRESSES
  AX = 0306h
Return: CF clear
  BX:CX -> procedure to switch from real to protected mode (see #03151)
  SI:(E)DI -> procedure to switch from protected to real mode
Notes:  16-bit programs should jump to SI:DI, 32-bit programs should use SI:EDI
  the caller must save and restore the state of the task with AX=0305h
  not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0305h

(Table 03151)
Values to JUMP at mode-switch procedures with:
  AX = new DS
  CX = new ES
  DX = new SS
  (E)BX = new (E)SP
  SI:(E)DI = new CS:(E)IP
Notes:  BP/EBP is preserved across the call, but AX/EAX, BX/EBX, CX/ECX,
```

```
    DX/EDX, SI/ESI, and DI/EDI will be undefined; FS and GS will be 0000h
  interrupts will stay disabled during the entire mode switch if they
    are disabled on entry to the mode-switch procedure
--------E-310400-----------------------------
INT 31 P - DPMI 0.9+ - GET DPMI VERSION
  AX = 0400h
Return: CF clear
  AH = major version of DPMI spec supported
  AL = two-digit minor version of DPMI spec supported
  BX = DPMI host flags (see #03152)
  CL = processor type (02h=80286, 03h=80386, 04h=80486)
  DH = curr value of virtual master interrupt controller base interrupt
  DL = curr value of virtual slave interrupt controller base interrupt
BUG:  Windows NT versions from the March 1993 beta to at least the Final
    release with fixes to CSD002 report version 0090h (0.144); this has
    reportedly been corrected in the Windows NT 3.5 beta
SeeAlso: AX=0401h,INT 21/AX=250Ch,INT 2F/AX=1687h,INT 4B/AX=8102h/DX=0000h
SeeAlso: INT 67/AX=DE0Ah

Bitfields for DPMI host flags:
Bit(s)  Description (Table 03152)
 0  running under an 80386 (32-bit) implementation
 1  processor returns to real mode for reflected interrupts instead of V86
    mode
 2  virtual memory supported
 3  reserved (undefined)
 4-15 reserved (zero)
--------E-310401-----------------------------
INT 31 P - DPMI 1.0+ - GET DPMI CAPABILITIES
  AX = 0401h
  ES:(E)DI -> 128-byte buffer for host description (see #03153)
Return: CF clear if successful
      AX = capabilities (see #03154)
      CX = reserved (00h)
      DX = reserved (00h)
      buffer filled
  CF set on error (DPMI 0.9 only)
SeeAlso: AX=0400h

Format of DPMI host description:
Offset  Size  Description (Table 03153)
```

```
 00h  BYTE  host major version number
 01h  BYTE  host minor version number
 02h 126 BYTEs  ASCIZ host vendor name

Bitfields for DPMI capabilities:
Bit(s)  Description (Table 03154)
 0  paged accessed/dirty supported (see AX=0506h,AX=0507h)
 1  exceptions restartability supported
 2  device mapping supported (see AX=0508h)
 3  conventional memory mapping supported (see AX=0509h)
 4  demand zero-fill supported
 5  write-protect client capability supported
 6  write-protect host capability supported
 7-15 reserved
--------E-310500-----------------------------
INT 31 P - DPMI 0.9+ - GET FREE MEMORY INFORMATION
  AX = 0500h
  ES:(E)DI -> buffer for memory information (see #03155)
Return: CF clear
Notes:  16-bit programs use ES:DI, 32-bit programs use ES:EDI
  this function must be considered advisory because other applications
    may affect the results at any time after the call
  fields not supported by the DPMI implementation are filled with
    FFFFFFFFh
  DPMI 1.0+ supports this function solely for backward compatibility; use
    AX=050Bh instead
  the limited DPMI host built into Phar Lap's 286|DOS-Extender v2.5 only
    returns the first field in the memory information record
SeeAlso: AX=0501h,AX=050Bh,AX=0604h

Format of DPMI memory information:
Offset  Size  Description (Table 03155)
 00h  DWORD largest available block in bytes
 04h  DWORD maximum unlocked page allocation
 08h  DWORD maximum locked page allocation
 0Ch  DWORD total linear address space in pages
 10h  DWORD total unlocked pages
 14h  DWORD free pages
 18h  DWORD total physical pages
 1Ch  DWORD free linear address space in pages
 20h  DWORD size of paging file/partition in pages
```

```
 24h 12 BYTEs reserved
--------E-310501-----------------------------
INT 31 P - DPMI 0.9+ - ALLOCATE MEMORY BLOCK
  AX = 0501h
  BX:CX = size in bytes
Return: CF clear if successful
      BX:CX = linear address of block
      SI:DI = memory block handle for resizing and freeing block
  CF set on error
      AX = error code (DPMI 1.0+) (8012h-8014h,8016h,8021h)(see #03143)
Notes:  no selectors are allocated
  the memory block is allocated unlocked (can be locked with AX=0600h)
  allocations are often page granular (see AX=0604h)
  under MS Windows 3.10 Enhanced mode with paging enabled, it is possible
    for this function to fail even if AX=0500h indicates that enough
    memory is available
SeeAlso: AX=0000h,AX=0100h,AX=0500h,AX=0502h,AX=0503h,AX=0504h,AX=0D00h
SeeAlso: INT 2F/AX=FB42h/BX=0002h
--------E-310502-----------------------------
INT 31 P - DPMI 0.9+ - FREE MEMORY BLOCK
  AX = 0502h
  SI:DI = handle of memory block
Return: CF clear if successful
  CF set on error
      AX = error code (DPMI 1.0+) (8023h) (see #03143)
Note: any selectors allocated for the memory block must also be freed,
    preferably before freeing the memory block
SeeAlso: AX=0001h,AX=0101h,AX=0501h,AX=0D01h
--------E-310503-----------------------------
INT 31 P - DPMI 0.9+ - RESIZE MEMORY BLOCK
  AX = 0503h
  BX:CX = new size in bytes (nonzero)
  SI:DI = handle of memory block
Return: CF clear if successful
      BX:CX = new linear address
      SI:DI = new handle of memory block
  CF set on error
      AX = error code (DPMI 1.0+) (8012h-8014h,8016h,8021h,8023h)
    (see #03143)
Notes:  any selectors pointing at the block must be updated
  the previous memory block handle becomes invalid
```

```
  an error is returned if the new size is 0
SeeAlso: AX=0102h,AX=0501h,AX=0505h
--------E-310504----------------------------
INT 31 P - DPMI 1.0+ - ALLOCATE LINEAR MEMORY BLOCK
  AX = 0504h
  EBX = page-aligned linear address of memory block (00000000h if any
        address is acceptable)
  ECX = size in bytes (nonzero)
  EDX = flags
      bit 0: set to create committed pages instead of uncommitted pages
      bits 1-31 reserved (0)
Return: CF clear if successful
      EBX = linear address of memory block
      ESI = memory block handle
  CF set on error
      AX = error code (8001h,8012h-8014h,8016h,8021h,8025h)(see #03143)
Note: only supported by 32-bit DPMI hosts, but may be used by 16-bit clients
SeeAlso: AX=0501h,AX=0505h
--------E-310505----------------------------
INT 31 P - DPMI 1.0+ - RESIZE LINEAR MEMORY BLOCK
  AX = 0505h
  ESI = memory block handle
  ECX = new size in bytes (nonzero)
  EDX = flags
      bit 0: create committed pages rather than uncommitted pages
      bit 1: segment descriptor update required
    ES:EBX -> buffer containing array of WORDs with selectors
    EDI = number of selectors in array
      bits 2-31 reserved (0)
Return: CF clear if successful
      EBX = new linear base address
      ESI = new memory block handle
  CF set on error
      AX = error code (8001h,8012h-8014h,8016h,8021h,8023h)(see #03143)
Notes:  only supported by 32-bit DPMI hosts, but may be used by 16-bit clients
  the old memory block handle becomes invalid
  if EDX bit 1 set and the block's base address is changed, DPMI updates
    all descriptors for selectors in the update buffer which fall within
    the memory block
SeeAlso: AX=0503h,AX=0504h
--------E-310506----------------------------
```

```
INT 31 P - DPMI 1.0+ - GET PAGE ATTRIBUTES
  AX = 0506h
  ESI = memory block handle
  EBX = offset in memory block of first page
  ECX = number of pages
  ES:EDX -> array of WORDs to hold page attributes (see #03156)
Return: CF clear if successful
      buffer filled
  CF set on error
      AX = error code (8001h,8023h,8025h) (see #03143)
Notes:  only supported by 32-bit DPMI hosts, but may be used by 16-bit clients
  if EBX is not page-aligned, it will be rounded down
SeeAlso: AX=0504h,AX=0507h,INT 21/AX=251Dh,INT 21/AX=EB00h

Bitfields for DPMI page attribute word:
Bit(s)  Description (Table 03156)
 0-2  page type
  000 uncommitted
  001 committed
  010 mapped (see AX=0508h,AX=0509h)
  other currently unused
 3  page is read/write rather than read-only
 4  accessed/dirty bits supplied in bits 5 and 6
 5  page has been accessed (only valid if bit 4 set)
 6  page has been written (only valid if bit 4 set)
 7-15 reserved (0)
--------E-310507-----------------------------
INT 31 P - DPMI 1.0+ - MODIFY PAGE ATTRIBUTES
  AX = 0507h
  ESI = memory block handle
  EBX = offset in memory block of first page
  ECX = number of pages
  ES:EDX -> array of WORDs with new page attributes (see #03156)
Return: CF clear if successful
  CF set on error
      AX = error code (8001h,8002h,8013h,8014h,8021h,8023h,8025h)
    (see #03143)
      ECX = number of pages which have been set
Notes:  only supported by 32-bit DPMI hosts, but may be used by 16-bit clients
  if EBX is not page-aligned, it will be rounded down
SeeAlso: AX=0504h,AX=0506h,INT 21/AX=251Eh
```

```
--------E-310508-----------------------------
INT 31 P - DPMI 1.0+ - MAP DEVICE IN MEMORY BLOCK
  AX = 0508h
  ESI = memory block handle
  EBX = page-aligned offset within memory block of page(s) to be mapped
  ECX = number of pages to map
  EDX = page-aligned physical address of device
Return: CF clear if successful
  CF set on error
      AX = error code (8001h,8003h,8023h,8025h) (see #03143)
Notes:  only supported by 32-bit DPMI hosts, but may be used by 16-bit clients
  support of this function is optional; hosts are also allowed to support
    the function for some devices but not others
SeeAlso: AX=0504h,AX=0509h,AX=0800h,AX=0801h
--------E-310509-----------------------------
INT 31 P - DPMI 1.0+ - MAP CONVENTIONAL MEMORY IN MEMORY BLOCK
  AX = 0509h
  ESI = memory block handle
  EBX = page-aligned offset within memory block of page(s) to map
  ECX = number of pages to map
  EDX = page-aligned linear address of conventional (below 1M) memory
Return: CF clear if successful
  CF set on error
      AX = error code (8001h,8003h,8023h,8025h) (see #03143)
Notes:  only supported by 32-bit DPMI hosts, but may be used by 16-bit clients
  support of this function is optional
SeeAlso: AX=0504h,AX=0508h,AX=0801h
--------E-31050A-----------------------------
INT 31 P - DPMI 1.0+ - GET MEMORY BLOCK SIZE AND BASE
  AX = 050Ah
  SI:DI = memory block handle
Return: CF clear if successful
      SI:DI = size in bytes
      BX:CX = base address
  CF set on error
      AX = error code (8023h) (see #03143)
SeeAlso: AX=0501h,AX=0504h
--------E-31050B-----------------------------
INT 31 P - DPMI 1.0+ - GET MEMORY INFORMATION
  AX = 050Bh
  ES:(E)DI -> 128-byte buffer for memory information (see #03157)
```

```
Return: CF clear if successful
  CF set on error (DPMI 0.9 only)
Note: 16-bit programs use ES:DI, 32-bit programs must use ES:EDI
SeeAlso: AX=0500h

Format of DPMI memory information:
Offset  Size  Description (Table 03157)
 00h  DWORD total allocated bytes of physical memory controlled by host
 04h  DWORD total allocated bytes of virtual memory controlled by host
 08h  DWORD total available bytes of virtual memory controlled by host
 0Ch  DWORD total allocated bytes of virtual memory for curr virtual mach
 10h  DWORD total available bytes of virtual memory for curr virtual mach
 14h  DWORD total allocated bytes of virtual memory for current client
 18h  DWORD total available bytes of virtual memory for current client
 1Ch  DWORD total locked bytes for current client
 20h  DWORD maximum locked bytes for current client
 24h  DWORD highest linear address available to current client
 28h  DWORD largest available memory block in bytes
 2Ch  DWORD minimum allocation unit in bytes
 30h  DWORD allocation alignment unit size in bytes
 34h 76 BYTEs reserved (00h)
--------E-310600-----------------------------
INT 31 P - DPMI 0.9+ - LOCK LINEAR REGION
  AX = 0600h
  BX:CX = starting linear address
  SI:DI = size of region in bytes
Return: CF clear if successful
  CF set on error
      none of the memory is locked
      AX = error code (DPMI 1.0+) (8013h,8017h,8025h) (see #03143)
Notes:  pages at beginning and end will be locked if the region overlaps them
  may be called multiple times for a given page; the DPMI host keeps a
    lock count for each page
SeeAlso: AX=0004h,AX=0601h,INT 21/AX=251Ah,INT 21/AX=EB06h
--------E-310601-----------------------------
INT 31 P - DPMI 0.9+ - UNLOCK LINEAR REGION
  AX = 0601h
  BX:CX = starting linear address
  SI:DI = size of region in bytes
Return: CF clear if successful
  CF set on error
```

```
      none of the memory is unlocked
      AX = error code (DPMI 1.0+) (8002h,8025h) (see #03143)
Notes:  pages at beginning and end will be unlocked if the region overlaps them
  memory whose lock count has not reached zero remains locked
SeeAlso: AX=0005h,AX=0600h,INT 21/AX=251Bh,INT 21/AX=EB07h
--------E-310602----------------------------
INT 31 P - DPMI 0.9+ - MARK REAL MODE REGION AS PAGEABLE
  AX = 0602h
  BX:CX = starting linear address
  SI:DI = size of region in bytes
Return: CF clear if successful
  CF set on error
      none of the memory is made pageable
      AX = error code (DPMI 1.0+) (8002h,8025h) (see #03143)
Notes:  must relock all unlocked real mode memory before terminating process
    for DPMI 0.9; DPMI 1.0+ automatically relocks real mode memory
  pages at beginning and end will be unlocked if the region overlaps them
  pageability of real mode pages is binary, not a count
SeeAlso: AX=0600h,AX=0603h
--------E-310603----------------------------
INT 31 P - DPMI 0.9+ - RELOCK REAL MODE REGION
  AX = 0603h
  BX:CX = starting linear address
  SI:DI = size of region in bytes
Return: CF clear if successful
  CF set on error
      none of the memory is relocked
      AX = error code (DPMI 1.0+) (8002h,8013h,8025h) (see #03143)
Notes:  pages at beginning and end will be relocked if the region overlaps them
  pageability of real mode pages is binary, not a count
SeeAlso: AX=0602h
--------E-310604----------------------------
INT 31 P - DPMI 0.9+ - GET PAGE SIZE
  AX = 0604h
Return: CF clear if successful
      BX:CX = page size in bytes
  CF set on error
      AX = error code (DPMI 1.0+) (see also #03143)
    8001h unsupported, 16-bit host
BUG:  the Borland C++ 3.1 DPMILOAD returns with CF clear but BX and CX
    unchanged
```

```
--------E-310700-----------------------------
INT 31 Pu - DPMI 0.9+ - MARK PAGES AS PAGING CANDIDATES
  AX = 0700h
  BX:CX = starting linear page number
  SI:DI = number of pages to mark as paging candidates
Return: ???
Note: although marked as reserved in versions 0.9 and 1.0 of the DPMI
    specification, this function is called by MS Windows TASKMAN,
    PROGMAN, and KERNEL
SeeAlso: AX=0701h,AX=0702h
--------E-310701-----------------------------
INT 31 Pu - DPMI 0.9+ - DISCARD PAGES
  AX = 0701h
  BX:CX = starting linear page number
  SI:DI = number of pages to discard
Return: ???
Note: although marked as reserved in versions 0.9 and 1.0 of the DPMI
    specification, this function is called by MS Windows TASKMAN,
    PROGMAN, and KERNEL
SeeAlso: AX=0700h,AX=0703h
--------E-310702-----------------------------
INT 31 P - DPMI 0.9+ - MARK PAGE AS DEMAND PAGING CANDIDATE
  AX = 0702h
  BX:CX = starting linear address
  SI:DI = number of bytes to mark as paging candidates
Return: CF clear if successful
  CF set on error
      AX = error code (DPMI 1.0+) (8025h) (see #03143)
Notes:  this function is advisory, and does not force immediate paging
  partial pages will not be discarded
SeeAlso: AX=0700h,AX=0703h
--------E-310703-----------------------------
INT 31 P - DPMI 0.9+ - DISCARD PAGE CONTENTS
  AX = 0703h
  BX:CX = starting linear address
  SI:DI = number of bytes to mark as discarded
Return: CF clear if successful
  CF set on error
      AX = error code (DPMI 1.0+) (8025h) (see #03143)
Notes:  this function is advisory, and may be ignored by DPMI implementations
  partial pages will not be discarded
```

```
SeeAlso: AX=0701h,AX=0702h
--------E-310800-----------------------------
INT 31 P - DPMI 0.9+ - PHYSICAL ADDRESS MAPPING
  AX = 0800h
  BX:CX = physical address (should be above 1 MB)
  SI:DI = size in bytes
Return: CF clear if successful
      BX:CX = linear address which maps the requested physical memory
  CF set on error
      AX = error code (DPMI 1.0+) (8003h,8021h) (see #03143)
Notes:  implementations may refuse this call because it can circumvent protects
  the caller must build an appropriate selector for the memory
  do not use for memory mapped in the first megabyte
SeeAlso: AX=0002h,AX=0508h,AX=0509h,AX=0801h,INT 21/AX=250Ah,INT 21/AX=EB05h
--------E-310801-----------------------------
INT 31 P - DPMI 1.0+ - FREE PHYSICAL ADDRESS MAPPING
  AX = 0801h
  BX:CX = linear address returned by AX=0800h
Return: CF clear if successful
  CF set on error
      AX = error code (8025h) (see #03143)
Note: should be called at end of access to device mapped with AX=0800h
SeeAlso: AX=0508h,AX=0509h,AX=0800h,INT 21/AX=EB03h
--------E-310900-----------------------------
INT 31 P - DPMI 0.9+ - GET AND DISABLE VIRTUAL INTERRUPT STATE
  AX = 0900h
Return: CF clear
  virtual interrupts disabled
  AL = previous interrupt state (00h disabled, 01h enabled)
  AH preserved
Notes:  the previous state may be restored simply by executing another INT 31
  a CLI instruction may be used if the previous state is unimportant,
    but should be assumed to be very slow due to trapping by the host
SeeAlso: AX=0901h,AX=0902h
--------E-310901-----------------------------
INT 31 P - DPMI 0.9+ - GET AND ENABLE VIRTUAL INTERRUPT STATE
  AX = 0901h
Return: CF clear
  virtual interrupts enabled
  AL = previous interrupt state (00h disabled, 01h enabled)
  AH preserved
```

```
Notes:  the previous state may be restored simply by executing another INT 31
  a STI instruction may be used if the previous state is unimportant,
    but should be assumed to be very slow due to trapping by the host
SeeAlso: AX=0900h,AX=0902h
--------E-310902-----------------------------
INT 31 P - DPMI 0.9+ - GET VIRTUAL INTERRUPT STATE
  AX = 0902h
Return: CF clear
  AL = current interrupt state (00h disabled, 01h enabled)
Note: should be used rather than PUSHF because that instruction yields the
    physical interrupt state rather than the per-client virtualized
    interrupt flag
SeeAlso: AX=0900h,AX=0901h
--------E-310A00-----------------------------
INT 31 P - DPMI 0.9+ - GET VENDOR SPECIFIC API ENTRY POINT
  AX = 0A00h
  DS:(E)SI -> case-sensitive ASCIZ vendor name or identifier
Return: CF clear if successful
      ES:(E)DI -> FAR extended API entry point
      DS, FS, GS, EAX, EBX, ECX, EDX, ESI, EBP destroyed
  CF set on error
      AX = error code (DPMI 1.0+) (8001h) (see #03143)
Notes:  extended API parameters are vendor-specific
  DPMI 1.0+ supports this function solely for backward compatibility; use
    INT 2F/AX=168Ah instead
  this function is not supported by MS Windows 3.10, BC++ 3.1 DPMILOAD,
    or QDPMI v1.0x; use INT 2F/AX=168Ah instead.  It is supported by
    386MAX v7.01.
SeeAlso: INT 2F/AX=168Ah
--------E-310B00-----------------------------
INT 31 P - DPMI 0.9+ - SET DEBUG WATCHPOINT
  AX = 0B00h
  BX:CX = linear address
  DL = size (1,2,4 bytes)
  DH = type (00h execute, 01h write, 02h read/write)
Return: CF clear if successful
      BX = watchpoint handle
  CF set on error
      AX = error code (DPMI 1.0+) (8016h,8021h,8025h) (see #03143)
SeeAlso: AX=0212h,AX=0601h
--------E-310B01-----------------------------
```

```
INT 31 P - DPMI 0.9+ - CLEAR DEBUG WATCHPOINT
  AX = 0B01h
  BX = watchpoint handle
Return: CF clear if successful
  CF set on error
      AX = error code (DPMI 1.0+) (8023h) (see #03143)
Note: the watchpoint handle is freed
SeeAlso: AX=0B00h
--------E-310B02-----------------------------
INT 31 P - DPMI 0.9+ - GET STATE OF DEBUG WATCHPOINT
  AX = 0B02h
  BX = watchpoint handle
Return: CF clear if successful
      AX = status flags
    bit 0: watch point has been executed since AX=0B00h or AX=0B03h
  CF set on error
      AX = error code (DPMI 1.0+) (8023h) (see #03143)
SeeAlso: AX=0B00h,AX=0B03h
--------E-310B03-----------------------------
INT 31 P - DPMI 0.9+ - RESET DEBUG WATCHPOINT
  AX = 0B03h
  BX = watchpoint handle
Return: CF clear if successful
  CF set on error
      AX = error code (DPMI 1.0+) (8023h) (see #03143)
SeeAlso: AX=0B02h
--------E-310C00-----------------------------
INT 31 P - DPMI 1.0+ - INSTALL RESIDENT HANDLER INIT CALLBACK
  AX = 0C00h
  ES:(E)DI -> resident service provider structure (see #03158)
Return: CF clear if successful
  CF set on error
      AX = error code (8015h,8021h,8025h) (see #03143 at AX=0000h)
Note: calling this function declares an intent to provide resident
    protected mode services after terminating with AX=0C01h
SeeAlso: AX=0303h,AX=0C01h

Format of DPMI resident service provider structure:
Offset  Size  Description (Table 03158)
 00h  QWORD descriptor for 16-bit data segment
 08h  QWORD descriptor for 16-bit code segment (zeros if not supported)
```

```
 10h  WORD  offset of 16-bit callback procedure
 12h  2 BYTEs reserved
 14h  QWORD descriptor for 32-bit data segment
 1Ch  QWORD descriptor for 32-bit code segment (zeros if not supported)
 24h  DWORD offset of 32-bit callback procedure
--------E-310C01-----------------------------
INT 31 P - DPMI 1.0+ - TERMINATE AND STAY RESIDENT
  AX = 0C01h
  BL = return code
  DX = number of paragraphs of DOS memory to reserve (0 or >= 6)
Return: never
Notes:  should only be used if the program will only provide services to
    other DPMI programs
  any protected mode memory remains allocated to the program unless
    explicitly freed before this call
  must first call AX=0C00h or program will simply be terminated
SeeAlso: AX=0C00h,INT 21/AH=31h
--------E-310D00-----------------------------
INT 31 P - DPMI 1.0+ - ALLOCATE SHARED MEMORY
  AX = 0D00h
  ES:(E)DI -> shared memory allocation request structure (see #03159)
Return: CF clear if successful
      request structure updated
  CF set on error
      AX = error code (8012h,8013h,8014h,8016h,8021h) (see #03143)
Note: first 16 bytes of memory block will be initialized to zeros on the
    first allocation
SeeAlso: AX=0501h,AX=0D01h,AX=0D02h

Format of DPMI shared memory allocation request structure:
Offset  Size  Description (Table 03159)
 00h  DWORD requested length of shared memory block in bytes
 04h  DWORD (ret) allocated length of block
 08h  DWORD (ret) shared memory handle
 0Ch  DWORD (ret) linear address of memory block
 10h  PWORD selector:offset32 of ASCIZ name for memory block
    (name max 128 bytes)
 16h  2 BYTEs reserved
 18h  4 BYTEs reserved (00h)
--------E-310D01-----------------------------
INT 31 P - DPMI 1.0+ - FREE SHARED MEMORY
```

```
  AX = 0D01h
  SI:DI = shared memory block handle
Return: CF clear if successful
  CF set on error
      AX = error code (8023h) (see #03143)
Notes:  handle becomes invalid after this call
  DPMI maintains separate global and virtual machine use counts for each
    shared memory block; when the global use counts reaches zero, the
    block is finally destroyed
SeeAlso: AX=0502h,AX=0D00h
--------E-310D02-----------------------------
INT 31 P - DPMI 1.0+ - SERIALIZE SHARED MEMORY
  AX = 0D02h
  SI:DI = shared memory block handle
  DX = flags
      bit 0: return immediately rather than suspending if serialization
      unavailable
      bit 1: shared rather than exclusive serialization
      bits 2-15 reserved (0)
Return: CF clear if successful
  CF set on error
      AX = error code (8004h,8005h,8017h-8019h,8023h) (see #03143)
Notes:  an exclusive serialization blocks any other serialization attempts for
    the same block by another virtual machine; a shared serialization
    blocks attempts at exclusive serialization by another virtual machine
  hosts are not required to detect deadlock
  a client's interrupt handler can cancel a serialization call which
    caused it to block by calling AX=0D03h
SeeAlso: AX=0D00h,AX=0D03h
--------E-310D03-----------------------------
INT 31 P - DPMI 1.0+ - FREE SERIALIZATION ON SHARED MEMORY
  AX = 0D03h
  SI:DI = shared memory block handle
  DX = flags
      bit 0: release shared serialization rather than exclusive serialztn
      bit 1: free pending serialization
      bits 2-15 reserved (0)
Return: CF clear if successful
  CF set on error
      AX = error code (8002h,8023h) (see #03143 at AX=0000h)
SeeAlso: AX=0D00h,AX=0D02h
```

```
--------E-310E00-----------------------------
INT 31 P - DPMI 1.0+ - GET COPROCESSOR STATUS
  AX = 0E00h
Return: CF clear
  AX = coprocessor status (see #03160)
Note: supported by 386MAX v6.01, which otherwise only supports DPMI 0.9
SeeAlso: AX=0E01h

Bitfields for DPMI coprocessor status:
Bit(s)  Description (Table 03160)
 0  numeric coprocessor enabled for current client
 1  client is emulating coprocessor
 2  numeric coprocessor is present
 3  host is emulating coprocessor instructions
 4-7  coprocessor type
  0000 none
  0010 80287
  0011 80387
  0100 80486 with numeric coprocessor
  other reserved
 8-15 not used
--------E-310E01-----------------------------
INT 31 P - DPMI 1.0+ - SET EMULATION
  AX = 0E01h
  BX = coprocessor flag bits (see #03161)
Return: CF clear if successful
  CF set on error
      AX = error code (8026h) (see #03143 at AX=0000h)
Note: supported by 386MAX v6.01, which otherwise only supports DPMI 0.9
SeeAlso: AX=0E00h

Bitfields for DPMI coprocessor flags:
Bit(s)  Description (Table 03161)
 0  enable numeric coprocessor for current client
 1  client will emulate coprocessor
 2-15 not used
--------E-3157------------------------------
INT 31 - Netroom3 DPMI.EXE v3.00 - ???
  AH = 57h
  AL = subfunction (at least 02h,03h,04h,05h,07h,08h,09h,0Ah)
  ???
```

```
Return: ???
SeeAlso: INT 2C/AX=0000h"Cloaking"
--------E-315702-----------------------------
INT 31 - Netroom3 DPMI.EXE v3.00 - SWITCH TO PROTECTED MODE
  AX = 5702h
  DX = PSP segment of caller
  STACK: WORD ???
         WORD flags (bit 0 set if 32-bit program)
Return: as for DPMI mode-switch entry point (see #02718 at INT 2F/AX=1687h)
Note: this function is called by the real-mode DPMI mode-switch entry point
SeeAlso: INT 2F/AX=1687h
--------E-31EE00------------------------
INT 31 - DOS32 v3.0+ - GET DOS32 VERSION AND SELECTOR VALUES
  AX = EE00h
Return: AL = minor version (BCD)
  AH = major version (BCD)
  DL = system type (1=raw DOS, 2=XMS, 4=VCPI, 8=DPMI)
  BX = selector of 4GB data segment with zero base address
Program: DOS32 is a 32 bit DOS extender by Adam Seychell.
SeeAlso: AX=EE02h
--------E-31EE02------------------------
INT 31 - DOS32 v3.0+ - GET DOS32 ADDRESS INFORMATION
  AX = EE02h
Return: AX = real-mode segment of temporary 8K buffer (v3.1+)
  EBX = 32bit linear address of the program segment
  EDX = Total size in bytes of the programs .EXE file after linking.
  ESI = offset address of PSP
  EDI = offset address of program environment
  ECX = offset address of the program's .EXE ASCIZ file name and path
SeeAlso: AX=EE00h,AX=EE20h
--------E-31EE10------------------------
INT 31 - DOS32 v3.2+ - SET UP A DOS32 LOADABLE LIBRARY
  AX = EE10h
  EDX -> library ASCIZ path\filename
  EBX = number of bytes to seek from beginning of file
Return: CF clear if successful
      EAX = size of memory block required to store library
      EBX = size of library file
  CF set on error
      AL = error code.
    01h error opening or reading file
```

```
    02h bad DOS32 library file
SeeAlso: AX=EE00h,AX=EE11h
--------E-31EE11-----------------------
INT 31 - DOS32 v3.2+ - LOAD LIBRARY FILE
  AX = EE11h
  EDX -> near pointer of memory block to store library
Return: CF clear if successful
      EDX = near pointer to the dynamic library public
  CF set on error
Note: must first successfuly call function AX=EE10h before calling this
    function
SeeAlso: AX=EE00h,AX=EE10h
--------E-31EE20-----------------------
INT 31 - DOS32 v3.0+ - GET REAL MODE CALL BACK ADDRESS WITH RETF STACK FRAME
  AX = EE20h
  ESI = offset of the real mode call back function
Return: CF clear if successful
      CX:DX = real mode address to call up to the protected mode
      procedure
  CF set on error
SeeAlso: AX=EE00h,AX=EE02h,AX=EE21h
--------E-31EE21-----------------------
INT 31 - DOS32 v3.0+ - GET REAL MODE CALL BACK ADDRESS WITH IRET STACK FRAME
  AX = EE21h
  ESI = offset of the real mode call back function
Return: CF clear if successful
      CX:DX = real mode address to call up to the protected mode
      procedure
  CF set on error
SeeAlso: AX=EE20h
--------E-31EE30-----------------------
INT 31 - DOS32 v3.0+ - TERMINATE AND STAY RESIDENT
  AX = EE30h
SeeAlso: AX=EE21h,AX=EE40h,INT 21/AH=31h
--------E-31EE40-----------------------
INT 31 - DOS32 v3.0+ - UNDO PREVIOUS MEMORY ALLOCATION or DMA BUFFER
  AX = EE40h
Return: CF clear if successful
  CF set on error
SeeAlso: AX=EE41h,AX=EE42h
--------E-31EE41-----------------------
```

```
INT 31 - DOS32 v3.0+ - ALLOCATE 16KB DMA BLOCK
  AX = EE41h
Return: CF clear if successful
      EBX -> 16KB DMA block (physical address)
      EDX -> 16KB DMA block (offset address)
  CF set on error
SeeAlso: AX=EE40h,AX=EE42h
--------E-31EE42------------------------
INT 31 - DOS32 v3.0+ - ALLOCATE MEMORY BLOCK
  AX = EE42h
  EDX = size in bytes
Return: CF clear if successful
      EAX = size in bytes
      EDX -> memory block
  CF set on error
Note: size is rounded off to the next 4KB boundary
SeeAlso: AX=EE40h,AX=EE41h
--------E-31FF00----------------------------
INT 31 P - CauseWay - "Info" - GET SYSTEM SELECTORS/FLAGS
  AX = FF00h
Return: AX = selector for flag address space (base 00000000h, limit 4GB)
  BX = selector for current PSP segment (limit 0100h)
  (E)CX = size of DOS transfer buffer (max 64K)
  DX = real-mode segment address of DOS transfer buffer
  ES:(E)SI = protected-mode address of DOS transfer buffer
  EDI = system flags (see #03162)
Program: CauseWay is a 386 DOS extender by Michael Devore and John Wildsmith
    for use with Watcom C++ or assembly language programs
Notes:  the entire transfer buffer can be addressed with a 16-bit offset in
    protected mode
  CauseWay always maps selector 0040h to the BIOS data segment at
    real-mode segment 0040h; when not running under a DPMI host, CauseWay
    also provides selectors A000h, B000h, and B800h mapped to video
    memory
SeeAlso: AX=FF25h

Bitfields for CauseWay system flags:
Bit(s)  Description (Table 03162)
 0  32-bit code
 1  virtual memory manage enabled
 3-2  mode: 00 raw extended memory, 01 VCPI, 10 DPMI
```

```
 4  DPMI available
 5  VCPI available
 6  no memory managers
 7  application descriptor table type: 0 = GDT, 1 = LDT
 14-8 reserved
 15 debugging engine present
--------E-31FF01-----------------------------
INT 31 P - CauseWay - "IntXX" - SIMULATE REAL-MODE INTERRUPT
  AX = FF01h
  BL = interrupt number
  ES:(E)DI -> real-mode register list (see #03148 at AX=0300h)
Return: register list updated
Note: CauseWay fills in the values for SS, SP, and FLAGS itself, and ignores
    the values specified for CS and IP
SeeAlso: AX=0300h,AX=FF02h
--------E-31FF02-----------------------------
INT 31 P - CauseWay - "FarCallReal" - SIMULATE REAL-MODE FAR CALL
  AX = FF02h
  ES:(E)DI -> real-mode register list (see #03148 at AX=0300h)
Return: register list updated
SeeAlso: AX=0301h,AX=FF01h
--------E-31FF03-----------------------------
INT 31 P - CauseWay - "GetSel" - ALLOCATE NEW SELECTOR
  AX = FF03h
Return: CF clear if successful
      BX = new selector
  CF set on error
Note: the new selector is initialized with a base address of 000000h, a limit
    of 0000h, and attributes read/write expand-up data
SeeAlso: AX=FF04h,AX=FF05h,AX=FF06h
--------E-31FF04-----------------------------
INT 31 P - CauseWay - "RelSel" - RELEASE A SELECTOR
  AX = FF04h
  BX = selector
Return: CF clear if successful
  CF set on error
SeeAlso: AX=FF03h,AX=FF06h
--------E-31FF05-----------------------------
INT 31 P - CauseWay - "CodeSel" - CONVERT SELECTOR TO EXECUTABLE CODE SELECTOR
  AX = FF05h
  BX = selector
```

```
  CL = default operation size (00h = 16-bit, 01h = 32-bit)
Return: CF clear if successful
  CF set on error
SeeAlso: AX=FF03h
--------E-31FF06----------------------------
INT 31 P - CauseWay - "AliasSel" - CREATE READ/WRITE DATA ALIAS SELECTOR
  AX = FF06h
  BX = original selector
Return: CF clear if successful
      AX = new data selector aliasing original selector
  CF set on error
SeeAlso: AX=FF03h,AX=FF04h
--------E-31FF07----------------------------
INT 31 P - CauseWay - "GetSelDet" - GET SELECTOR BASE AND LIMIT
  AX = FF07h
  BX = selector
Return: CF clear if successful
      CX:DX = base address
      SI:DI = limit (bytes)
  CF set on error
SeeAlso: AX=FF08h,AX=FF09h
--------E-31FF08----------------------------
INT 31 P - CauseWay - "GetSelDet32" - GET SELECTOR BASE AND LIMIT (32-bit)
  AX = FF08h
Return: CF clear if successful
      EDX = base address
      ECX = limit (bytes)
  CF set on error
SeeAlso: AX=FF07h,AX=FF0Ah
--------E-31FF09----------------------------
INT 31 P - CauseWay - "SetSelDet" - SET SELECTOR BASE AND LIMIT
  AX = FF09h
  BX = selector
  CX:DX = new base address
  SI:DI = new byte-granular limit
Return: CF clear if successful
  CF set on error
SeeAlso: AX=FF07h,AX=FF0Ah
--------E-31FF0A----------------------------
INT 31 P - CauseWay - "SetSelDet32" - SET SELECTOR BASE AND LIMIT (32-bit)
  AX = FF0Ah
```

```
  BX = selector
  EDX = new base address
  ECX = new byte-granular limit
Return: CF clear if successful
  CF set on error
SeeAlso: AX=FF08h,AX=FF09h
--------E-31FF0B-----------------------------
INT 31 P - CauseWay - "GetMem" - ALLOCATE BLOCK OF MEMORY
  AX = FF0Bh
  CX:DX = size in bytes (FFFFh:FFFFh to get size of largest available)
Return: CF clear if successful
      BX = selector for accessing block if requested size not FFFFh:FFFFh
      CX:DX = size of largest available block if requested FFFFh:FFFFh
  CF set on error
SeeAlso: AX=FF0Ch,AX=FF0Dh,AX=FF0Fh,AX=FF10h,AX=FF2Ch
--------E-31FF0C-----------------------------
INT 31 P - CauseWay - "GetMem32" - ALLOCATE BLOCK OF MEMORY (32-bit)
  AX = FF0Ch
  ECX = size in bytes (FFFFFFFFh to get size of largest available block)
Return: CF clear if successful
      BX = selector for accessing block if requested size not FFFFh:FFFFh
      ECX = size of largest available block if requested FFFFh:FFFFh
  CF set on error
SeeAlso: AX=FF0Bh,AX=FF0Eh,AX=FF0Fh,AX=FF11h
--------E-31FF0D-----------------------------
INT 31 P - CauseWay - "ResMem" - RESIZE MEMORY BLOCK
  AX = FF0Dh
  BX = selector for block to be resized
  CX:DX = new size in bytes
Return: CF clear if successful
  CF set on error
Note: the memory block may have to be copied to another location in order
    to satisfy the requested new size, in which case the base address
    of the selector is updated
SeeAlso: AX=FF0Bh,AX=FF0Eh,AX=FF12h
--------E-31FF0E-----------------------------
INT 31 P - CauseWay - "ResMem32" - RESIZE MEMORY BLOCK (32-bit)
  AX = FF0Eh
  BX = selector for block to be resized
  ECX = new size in bytes
Return: CF clear if successful
```

```
  CF set on error
Note: the memory block may have to be copied to another location in order
    to satisfy the requested new size, in which case the base address
    of the selector is updated
SeeAlso: AX=FF0Ch,AX=FF0Dh,AX=FF13h
--------E-31FF0F----------------------------
INT 31 P - CauseWay - "RelMem" - RELEASE PREVIOUSLY ALLOCATED MEMORY
  AX = FF0Fh
  BX = selector for block to be released
Return: CF clear if successful
  CF set on error
SeeAlso: AX=FF0Bh,AX=FF0Ch,AX=FF14h
--------E-31FF10----------------------------
INT 31 P - CauseWay - "GetMemLinear" - ALLOCATE MEMORY WITHOUT SELECTOR
  AX = FF10h
  CX:DX = size of block in bytes
Return: CF clear if successful
      SI:DI = linear address of allocated block
  CF set on error
Note: this function may return addresses above 16M
SeeAlso: AX=FF0Bh,AX=FF11h,AX=FF12h,AX=FF14h
--------E-31FF11----------------------------
INT 31 P - CauseWay - "GetMemLinear32" - ALLOCATE MEMORY WITHOUT SELECTOR
  AX = FF11h
  ECX = size of block in bytes
Return: CF clear if successful
      ESI = linear address of allocated block
  CF set on error
Note: this function may return addresses above 16M
SeeAlso: AX=FF0Ch,AX=FF10h,AX=FF13h,AX=FF14h
--------E-31FF12----------------------------
INT 31 P - CauseWay - "ResMemLinear" - RESIZE LINEAR MEMORY BLOCK
  AX = FF12h
  CX:DX = new size in bytes
  SI:DI = linear address of block to be resized
Return: CF clear if successful
     SI:DI = new linear address of block
  CF set on error
SeeAlso: AX=FF0Dh,AX=FF10h,AX=FF13h
--------E-31FF13----------------------------
INT 31 P - CauseWay - "ResMemLinear32" - RESIZE LINEAR MEMORY BLOCK (32-bit)
```

```
  AX = FF13h
  ECX = new size in bytes
  ESI = linear address of block to be resized
Return: CF clear if successful
     ESI = new linear address of block
  CF set on error
SeeAlso: AX=FF0Eh,AX=FF11h,AX=FF12h
--------E-31FF14----------------------------
INT 31 P - CauseWay - "RelMemLinear" - RELEASE LINEAR MEMORY BLOCK
  AX = FF14h
  SI:DI = linear address of block to be released
Return: CF clear if successful
  CF set on error
SeeAlso: AX=FF0Fh,AX=FF10h,AX=FF15h
--------E-31FF15----------------------------
INT 31 P - CauseWay - "RelMemLinear32" - RELEASE LINEAR MEMORY BLOCK (32-bit)
  AX = FF15h
  ESI = linear address of block to be released
Return: CF clear if successful
  CF set on error
SeeAlso: AX=FF0Fh,AX=FF11h,AX=FF15h
--------E-31FF16----------------------------
INT 31 P - CauseWay - "GetMemNear" - ALLOCATE APPLICATION-RELATIVE MEMORY
  AX = FF16h
  EBX = size in bytes of block to allocate
Return: CF clear if successful
     ESI = application-relative linear address of allocated block
  CF set on error
SeeAlso: AX=FF17h,AX=FF18h,AX=FF19h
--------E-31FF17----------------------------
INT 31 P - CauseWay - "ResMemNear" - RESIZE APPLICATION-RELATIVE MEMORY BLOCK
  AX = FF17h
  EBX = size in bytes of block to allocate
  ESI = application-relative linear address of block
Return: CF clear if successful
      ESI = new application-relative linear address of block
  CF set on error
SeeAlso: AX=FF16h,AX=FF18h,AX=FF19h
--------E-31FF18----------------------------
INT 31 P - CauseWay - "RelMemNear" - RELEASE APPLICATION-RELATIVE MEMORY BLOCK
  AX = FF18h
```

```
  ESI = application-relative linear address of block
Return: CF clear if successful
  CF set on error
SeeAlso: AX=FF16h,AX=FF17h,AX=FF19h
--------E-31FF19----------------------------
INT 31 P - CauseWay - "Linear2Near" - CONVERT LINEAR TO APP-RELATIVE ADDRESS
  AX = FF19h
  ESI = linear address
Return: ESI = application-relative linear address
SeeAlso: AX=FF1Ah
--------E-31FF1A----------------------------
INT 31 P - CauseWay - "Near2Linear" - CONVERT APP-RELATIVE TO LINEAR ADDRESS
  AX = FF1Ah
  ESI = application-relative linear address
Return: ESI = linear address
SeeAlso: AX=FF19h
--------E-31FF1B----------------------------
INT 31 P - CauseWay - "LockMem" - LOCK REGION OF MEMORY
  AX = FF1Bh
  BX:CX = starting linear address
  SI:DI = size of region in bytes
Return: CF clear if successful
  CF set on error
Desc: prevent memory from being swapped out by the virtual memory manager
Note: locks are applied on 4K page boundaries, so memory above and below
    the specified region may be locked as well
SeeAlso: AX=FF1Ch,AX=FF1Dh
--------E-31FF1C----------------------------
INT 31 P - CauseWay - "LockMem32" - LOCK REGION OF MEMORY (32-bit)
  AX = FF1Ch
  ESI = starting linear address
  ECX = size of region in bytes
Return: CF clear if successful
  CF set on error
Desc: prevent memory from being swapped out by the virtual memory manager
Note: locks are applied on 4K page boundaries, so memory above and below
    the specified region may be locked as well
SeeAlso: AX=FF1Bh,AX=FF1Eh,AX=FF1Fh
--------E-31FF1D----------------------------
INT 31 P - CauseWay - "UnLockMem" - UNLOCK REGION OF MEMORY
  AX = FF1Dh
```

```
  BX:CX = starting linear address
  SI:DI = size of region in bytes
Return: CF clear if successful
  CF set on error
Desc: allow memory to be swapped out by the virtual memory manager
Note: locks are applied on 4K page boundaries, so memory above and below
    the specified region may be unlocked as well
SeeAlso: AX=FF1Bh,AX=FF1Eh
--------E-31FF1E-----------------------------
INT 31 P - CauseWay - "UnLockMem32" - UNLOCK REGION OF MEMORY (32-bit)
  AX = FF1Eh
  ESI = starting linear address
  ECX = size of region in bytes
Return: CF clear if successful
  CF set on error
Desc: allow memory to be swapped out by the virtual memory manager
Note: locks are applied on 4K page boundaries, so memory above and below
    the specified region may be unlocked as well
SeeAlso: AX=FF1Ch,AX=FF1Dh,AX=FF20h
--------E-31FF1F-----------------------------
INT 31 P - CauseWay - "LockMemNear" - LOCK APPLICATION-RELATIVE MEMORY REGION
  AX = FF1Fh
  ESI = starting application-relative linear address
  EBX = size of region in bytes
Return: CF clear if successful
  CF set on error
Desc: prevent memory from being swapped out by the virtual memory manager
Note: locks are applied on 4K page boundaries, so memory above and below
    the specified region may be locked as well
SeeAlso: AX=FF1Ch,AX=FF20h
--------E-31FF20-----------------------------
INT 31 P - CauseWay - "UnLockMemNear" - UNLOCK APP-RELATIVE MEMORY REGION
  AX = FF20h
  ESI = starting application-relative linear address
  ECX = size of region in bytes
Return: CF clear if successful
  CF set on error
Desc: allow memory to be swapped out by the virtual memory manager
Note: locks are applied on 4K page boundaries, so memory above and below
    the specified region may be unlocked as well
SeeAlso: AX=FF1Eh,AX=FF1Fh
```

```
--------E-31FF21-----------------------------
INT 31 P - CauseWay - "GetMemDOS" - ALLOCATE CONVENTIONAL MEMORY
  AX = FF21h
  BX = number of paragraphs to allocate
Return: CF clear if successful
      AX = real-mode segment of allocated block
      DX = initial selector for block
  CF set on error
      AX = DOS error code (see #01680 at INT 21/AH=59h/BX=0000h)
      BX = size of largest available block
Note: if the requested size is greater than 64K, contiguous selectors will
    be allocated, one for each 64K; for 32-bit applications, the first
    selector's limit will be set to the size of the entire block, while
    that selector will have a limit of 64K for 16-bit applications.  All
    selectors after the first one have a limit of 64K except the final
    one
SeeAlso: AX=FF22h,AX=FF23h,INT 21/AH=48h
--------E-31FF22-----------------------------
INT 31 P - CauseWay - "ResMemDOS" - RESIZE CONVENTIONAL MEMORY BLOCK
  AX = FF22h
  BX = new size in paragraphs
  DX = initial selector for block
Return: CF clear if successful
  CF set on error
      AX = DOS error code (see #01680 at INT 21/AH=59h/BX=0000h)
      BX = maximum number of paragraphs available
Note: this function will also fail if the block is expanded past a 64K
    boundary and the next selector is already in use
SeeAlso: AX=FF21h,AX=FF23h,INT 21/AH=49h
--------E-31FF23-----------------------------
INT 31 P - CauseWay - "RelMemDOS" - RELEASE CONVENTIONAL MEMORY BLOCK
  AX = FF23h
  DX = initial selector for block
Return: CF clear if successful
  CF set on error
      AX = DOS error code (see #01680 at INT 21/AH=59h/BX=0000h)
Note: all descriptors allocated for the block are freed
SeeAlso: AX=FF21h,AX=FF22h,INT 21/AH=4Ah
--------E-31FF24-----------------------------
INT 31 P - CauseWay - "ExecOverlay" - LOAD AND OPTIONALLY EXECUTE APP CODE
  AX = FF24h
```

```
  EBX = flags
      bit 0: don't execute (overlay only)
      bit 1: don't preserve relocation information
  ES:EDX -> filename (see also #03163)
  FS:ESI -> commandline (ESI = 00000000h if no commandline)
  GS:EDI -> name (CW's /o option)
Return: CF clear if successful
      ---EBX bit 0 set---
      CX:EDX = program entry point (CS:EIP)
      SI = segment of PSP
      ---EBX bit 1 set---
      BX:EAX = initial SS:ESP
      EDI high word = base segment
      EDI low word = number of segments
      EBP = start of segment definitions
  CF set on error
      AX = error code
    0001h DOS file access error
    0002h not a CuaseWay 3P file
    0003h not enough memory
SeeAlso: AX=FF2Ah,INT 21/AH=4Bh

Format of CauseWay executable:
Offset  Size  Description (Table 03163)
 00h  2 BYTEs signature "3P"
 02h  DWORD size of header data in bytes
 06h  DWORD size of EXE image data in bytes
 0Ah  DWORD number of bytes of program memory required
 0Eh  WORD  number of segment definitions (see #03165)
 10h  DWORD number of relocation table entries
 14h  DWORD offset of program entry point
 18h  WORD  segment list entry number for entry point's CS
 1Ah  DWORD initial ESP
 1Eh  WORD  segment list entry number for initial SS
 20h  DWORD control flags (see #03164)
 24h  DWORD automatic stack size in bytes if ESP entry = 00000000h
 28h  BYTE  length of name (name follows program image)
 29h 23 BYTEs reserved
SeeAlso: #01594,#01609 at INT 21/AH=4Bh

Bitfields for CauseWay executable control flags:
```

```
Bit(s)  Description (Table 03164)
 0  16-bit interrupt stack frame
 7  descriptor table type (0 = GDT, 1 = LDT)
 14 16-bit default data size
 31 compressed EXE image
Note: bits 0 and 14 should always be equal
SeeAlso: #03163

Format of CauseWay segment definition [array]:
Offset  Size  Description (Table 03165)
 00h  DWORD start offset within program image
 04h  DWORD length and type
    bits 0-19: length
    bit 20:    granularity (if set, length is in 4K pages)
    bits 21-24: type
      0000 code
      0001 read/write data
      0010 stack
      0011 read-only data
    bit 25: force segment descriptor's D bit to 0
    bit 26: force segmetn descriptor's D bit to 1
SeeAlso: #03163
--------E-31FF25-----------------------------
INT 31 P - CauseWay - "GetDOSTrans" - GET DOS TRANSFER BUFFER
  AX = FF25h
Return: BX = real-mode segment of transfer buffer
  ECX = transfer buffer size
  DX = protected-mode selector for transfer buffer
Note: the default buffer size of 8K is sufficient for most applications,
    but an application performing large amounts of file I/O may benefit
    from allocating its own, larger buffer
SeeAlso: AX=FF00h,AX=FF26h
--------E-31FF26-----------------------------
INT 31 P - CauseWay - "SetDOSTrans" - SET DOS TRANSFER BUFFER
  AX = FF26h
  BX = real-mode segment of new transfer buffer
  ECX = new transfer buffer size
  DX = protected-mode selector for new transfer buffer
Return: nothing
Note: the specific buffer must be located in conventional memory; only the
    first 64K will be used
```

```
SeeAlso: AX=FF25h
--------E-31FF27-----------------------------
INT 31 P - CauseWay v1.3 - "GetMCBSize" - GET CURRENT MCB ALLOCATION BLOCK SIZE
  AX = FF27h
Return: ECX = current threshold
Desc: determine the memory allocation size below which CauseWay will use
    internal MCB chains rather than allocating full 4K pages via DPMI
SeeAlso: AX=FF00h,AX=FF28h
--------E-31FF28-----------------------------
INT 31 P - CauseWay v1.3 - "SetMCBSize" - SET MCB MEMORY ALLOCATION BLOCK SIZE
  AX = FF28h
  ECX = new threshold (0000000h to disable MCB memory allocation system)
Return: CF clear if successful
      threshold unchanged (default 16K)
  CF set on error (threshold > 64K)
Desc: specify the memory allocation size below which CauseWay will use
    internal MCB chains rather than allocating full 4K pages via DPMI
Note: the specified threshold will be rounded up to the next higher multiple
    of 4K
SeeAlso: AX=FF00h,AX=FF27h
--------E-31FF29-----------------------------
INT 31 P - CauseWay v1.3 - "GetSels" - ALLOCATE MULTIPLE SELECTORS
  AX = FF29h
  CX = number of selectors to allocate
Return: BX = base selector
Desc: allocate multiple contiguous selectors, initializing each to have a
    base address and limit of zero
SeeAlso: AX=FF00h,AX=FF04h
--------E-31FF2A-----------------------------
INT 31 P - CauseWay v1.3 - "cwLoad" - LOAD ANOTHER CAUSEWAY PROGRAM AS OVERLAY
  AX = FF2Ah
  DS:EDX -> filename
Return: CF clear if successful
      CX:EDX = CS:EIP of entry point
      BX:EAX = initial SS:ESP for program
      SI = PSP for overlay program
  CF set on error
      AX = error code (01h file error, 02h not a 3P file, 03h no memory)
Note: the returned PSP can be given to "RelMem" (AX=FF0Fh) to release the
    overlay's memory and selectors; only selectors and memory allocated
    during loading will be freed by RelMem unless one switches PSPs with
```

```
    INT 21/AH=50h
SeeAlso: AX=FF00h,AX=FF24h
--------E-31FF2B-----------------------------
INT 31 P - CauseWay v1.3 - "cwcInfo" - VALIDATE AND GET SIZE OF CWC FILE
  AX = FF2Bh
  BX = file handle for CWC-compressed file
Return: CF clear if successful
      ECX = expanded file size
  CF set on error (not a CWC-compressed file)
SeeAlso: AX=FF00h,AX=FFFBh
--------E-31FF2C-----------------------------
INT 31 P - CauseWay v1.3 - "GetMemSO" - ALLOCATE MEMORY AND RETURN SEL:OFFSET
  AX = FF2Ch
  CX:DX = block size in bytes
Return: CF clear if successful
     SI:DI = selector:offset of allocated memory
  CF set on error
Note: unlike "GetMem" (AX=FF0Bh), this function will reuse selectors until a
    segment is full, rather than allocating a new selector for each
    memory block
SeeAlso: AX=FF00h,AX=FF2Dh,AX=FF2Eh
--------E-31FF2D-----------------------------
INT 31 P - CauseWay v1.3 - "ResMemSO" - RESIZE SELECTOR:OFFSET MEMORY BLOCK
  AX = FF2Dh
  SI:DI = selector:offset for memory block
  CX:DX = new size of block
Return: CF clear if successful
     SI:DI = new selector:offset for memory block
  CF set on error
SeeAlso: AX=FF00h,AX=FF2Ch,AX=FF2Eh
--------E-31FF2E-----------------------------
INT 31 P - CauseWay v1.3 - "RelMemSO" - RELEASE SELECTOR:OFFSET MEMORY BLOCK
  AX = FF2Eh
  SI:DI = selector:offset for memory block
SeeAlso: AX=FF00h
--------E-31FFFB-----------------------------
INT 31 P - Causeway v1.3 - "cwcLoad" - LOAD/EXPAND CWC-COMPRESSED FILE
  AX = FFFBh
  BX = source file handle
  ES:EDI -> memory buffer into which to expand file
Return: CF clear if successful
```

```
      ECX = expanded data length
  CF set on error
      EAX = error code (01h file error, 02h bad data, 03h not CWC file)
Note: the provided file may consist of the concatenation of several CWC
    files; the one beginning at the current file position will be
    expanded
SeeAlso: AX=FF2Bh
--------v-32---------------------------------
INT 32 - VIRUS - "Tiny" Viruses - ORIGINAL INT 21h VECTOR
SeeAlso: INT 21/AX=FFFFh"VIRUS",INT 31"VIRUS",INT 44"VIRUS"
--------v-32---------------------------------
INT 32 - VIRUS - "Plovdiv 1.3"/"Damage 1.3" - ORIGINAL INT 21h VECTOR
SeeAlso: INT 31"VIRUS",INT 9E"VIRUS"
--------y-326E-------------------------------
INT 32 - NOISE.SYS 0.53 - API
  AH = 6Eh (function ID)
  AL = subfunction (see INT 32/AX=6E00h)
Return: CF set on error
      AL = error code (see #03166)
  CF clear if successful
Notes:  INT 32 is only a proposed interface for NOISE.SYS. Use the IOCTL READ
    from the RANDOM device to determine the interrupt and function ID
    used by the driver, since future versions may use the Alternate
    Multiplex Interrupt (AMIS) at INT 2Dh.
  the beta v0.51 had a substantially different API on INT 32/AH=6Eh

(Table 03166)
Values for NOISE.SYS error codes:
 00h  subfunction not supported
 FBh  random pool is empty
 FCh  quality of sample is too low
 FDh  too many processes using the API or driver
 FEh  subfunction is disabled in the current build
 FFh  successful
--------y-326E00-----------------------------
INT 32 - NOISE.SYS v0.53+ - INSTALLATION CHECK
  AX = 6E00h
Return: AL = installation status
      00h not installed
      FFh installed
    CX = version (ie, 0123h = Version 1.2.3)
```

```
    DX:DI -> signature string
--------y-326E01-----------------------------
INT 32 - NOISE.SYS v0.53+ - GET ENTRY POINT
  AX = 6E01h
Return: AL = FFh
  DX:DI -> far call hook
----------326E04-----------------------------
INT 32 - NOISE.SYS v0.55+ - GET INTERRUPT HOOK LIST
  AX = 6E04h
Return: AL = status
      00h = unimplemented
      04h = DX:BX -> interrupt hook list
      FEh = subfunction disabled
Note: the hook list array ends with API interrupt (usually 32h, although
    it will differ if the API is installed at another interrupt)
----------326E06-----------------------------
INT 32 - NOISE.SYS v0.55+ - GET DEVICE DRIVER HEADER
  AX = 6E06h
Return: AL = number of device drivers in NOISE.SYS chain
      02h = default (for RANDOM and URANDOM devices)
  AH = AMIS device driver flags (set to 00h for now)
  DX:BX -> first device in chain (see #01646)
SeeAlso: INT 2D/AL=06h
--------y-326E10-----------------------------
INT 32 - NOISE.SYS v0.53+ - STATUS CHECK
  AX = 6E10h
Return: CF set on error
      AL = error code (FDh) (see #03166)
  CF clear if successful
      AL = status
    FFh successful
      BH = number of processes using the API
      CX = number of random bytes waiting
      DX = maximum possible bytes waiting
    (if CX=DX, the pool is full)
Note: this subfunction is a convenient way to check the driver if any
    fresh bytes are waiting in the output pool.
SeeAlso: INT 32/AH=6Eh,AX=6E00h,AX=6E11h
--------y-326E11-----------------------------
INT 32 - NOISE.SYS v0.53+ - GET ENTROPY ESTIMATE
  AX = 6E11h
```

```
Return: CF set on error
      AL = error code (00h,FDh,FEh) (see #03166)
  CF clear if successful
      EBX = estimated bit count (refer to note below)
      CL = FRACBITS (number of fractional bits)
      EDX = low 32-bits of total number of samples added
Note:  the estimated number of fresh random bits is equal to
    (EAX >> FRACBITS) + ((EAX & ((1 << FRACBITS)-1) / (1 << FRACBITS))
SeeAlso: AH=6Eh,AX=6E00h
--------y-326E12-----------------------------
INT 32 - NOISE.SYS v0.53+ - ADD SAMPLE FROM FAST TIMER
  AX = 6E12h
Return: CF set on error
      AL = error code (FCh,FDh,FEh) (see #03166)
  CF clear if successful
      CX = number of random bytes waiting
Note: subfunctions 12h and 13h are meant for applications or devices
    which are able to gather entropy from other sources which are
    not polled by NOISE.SYS (for example, a communications driver
    could use this call to sample packet arrival times).
SeeAlso: AX=6E00h,AX=6E10h,AX=6E13h
--------y-326E13-----------------------------
INT 32 - NOISE.SYS v0.53+ - ADD 16-BIT SAMPLE TO RANDOM POOL
  AX = 6E13h
  DX = sample
Return: CF set on error
      AL = error code (FCh,FEh) (see #03166)
  CF clear if successful
      CX = number of random bytes waiting
SeeAlso: AX=6E00h,AX=6E11h
--------y-326E14-----------------------------
INT 32 - NOISE.SYS v0.53+ - GET FLAGS
  AX = 6E14h
Return: BX = flags (see #03167)
  CX = mask of settable flags in BX
SeeAlso: AX=6E00h,AX=6E15h

Bitfields for NOISE.SYS flags:
Bit(s)  Description (Table 03167)
 0  MS Windows active
 1-5  reserved
```

```
 6  clock drift sampling
 7  video retrace drift sampling
 8  network access sampling (not implemented yet in 0.53)
 9  CD-ROM access sampling (not implemented yet in 0.53)
 10 DOS spinner
 11 DOS process start/end and miscellaneous process activity sampling
 12 mouse movement/button sampling
 13 disk sampling (INT 13)
 14 keystroke timings
 15 reserved for hardware RNG
--------y-326E15-----------------------------
INT 32 - NOISE.SYS v0.53+ - SET FLAGS
  AX = 6E15h
  BX = flags (see #03167)
Return: BX = new flags
Note: flags which AX=6E14h indicates are not settable should be masked off
    by ANDing with the CX returned by AX=6E14h
SeeAlso: AX=6E00h,AX=6E14h
--------y-326E16-----------------------------
INT 32 - NOISE.SYS v0.53+ - READ URANDOM BYTES
  AX = 6E16h
  CX = number of bytes
  ES:DI -> buffer
Return: CF set on error
      AL = error code (FDh,FEh) (see #03166)
  CF clear if successful
      CX = number of random bytes read
SeeAlso: AX=6E00h,AX=6E12h,AX=6E17h
--------y-326E17-----------------------------
INT 32 - NOISE.SYS v0.53+ - READ RANDOM BYTES
  AX = 6E17h
  CX = number of bytes
  ES:DI -> buffer
Return: CF set on error
      AL = error code (FBh,FDh,FEh) (see #03166)
  CF clear if successful
      CX = number of random bytes read
SeeAlso: AX=6E00h,AX=6E16h
----------326E18-----------------------------
INT 32 - NOISE.SYS v0.6+ - READ CONTROL RECORD
  AX = 6E18h
```

```
  CX = buffer size
  ES:DI -> buffer
Return: AL = status
      00h unimplemented (before v0.6)
      FEh subfunction is disabled
      FFh successful
    CX = number of bytes read
Note: the control record corresponds to the IOCTL Read record for the RANDOM
    device
----------326E-------------------------------
INT 32 - NOISE.SYS - RESERVED FOR FUTURE USE
  AH = 6Eh
  AL = 19h to 3Fh
Return: AL = 00h
Note: these functions are reserved for future use; user additions to the
    the driver should use subfunctions 40h to FFh.
--------M-330000-----------------------------
INT 33 - MS MOUSE - RESET DRIVER AND READ STATUS
  AX = 0000h
Return: AX = status
      0000h hardware/driver not installed
      FFFFh hardware/driver installed
  BX = number of buttons
      0000h other than two
      0002h two buttons (many drivers)
      0003h Mouse Systems/Logitech three-button mouse
      FFFFh two buttons
Notes:  since INT 33 might be uninitialized on old machines, the caller
    should first check that INT 33 is neither 0000h:0000h nor points at
    an IRET instruction (BYTE CFh) before calling this API
  to use mouse on a Hercules-compatible monographics card in graphics
    mode, you must first set 0040h:0049h to 6 for page 0 or 5 for page 1,
    and then call this function.  Logitech drivers v5.01 and v6.00
    reportedly do not correctly use Hercules graphics in dual-monitor
    systems, while version 4.10 does.
  the Logitech mouse driver contains the signature string "LOGITECH"
    three bytes past the interrupt handler; many of the Logitech mouse
    utilities check for this signature.
  Logitech MouseWare v6.30 reportedly does not support CGA video modes
    if no CGA is present when it is started and the video board is
    later switched into CGA emulation
```

```
SeeAlso: AX=0011h,AX=0021h,AX=002Fh,INT 62/AX=007Ah,INT 74
--------M-330001-----------------------------
INT 33 - MS MOUSE v1.0+ - SHOW MOUSE CURSOR
  AX = 0001h
SeeAlso: AX=0002h,INT 16/AX=FFFEh,INT 62/AX=007Bh,INT 6F/AH=06h"F_TRACK_ON"
--------M-330002-----------------------------
INT 33 - MS MOUSE v1.0+ - HIDE MOUSE CURSOR
  AX = 0002h
Note: multiple calls to hide the cursor will require multiple calls to
    function 01h to unhide it.
SeeAlso: AX=0001h,AX=0010h,INT 16/AX=FFFFh,INT 62/AX=007Bh
SeeAlso: INT 6F/AH=08h"F_TRACK_OFF"
--------M-330003-----------------------------
INT 33 - MS MOUSE v1.0+ - RETURN POSITION AND BUTTON STATUS
  AX = 0003h
Return: BX = button status (see #03168)
  CX = column
  DX = row
Note: in text modes, all coordinates are specified as multiples of the cell
    size, typically 8x8 pixels
SeeAlso: AX=0004h,AX=000Bh,INT 2F/AX=D000h"ZWmous"

Bitfields for mouse button status:
Bit(s)  Description (Table 03168)
 0  left button pressed if 1
 1  right button pressed if 1
 2  middle button pressed if 1 (Mouse Systems/Logitech/Genius)
--------M-330004-----------------------------
INT 33 - MS MOUSE v1.0+ - POSITION MOUSE CURSOR
  AX = 0004h
  CX = column
  DX = row
Note: the row and column are truncated to the next lower multiple of the cell
    size (typically 8x8 in text modes); however, some versions of the
    Microsoft documentation incorrectly state that the coordinates are
    rounded
SeeAlso: AX=0003h,INT 62/AX=0081h,INT 6F/AH=10h"F_PUT_SPRITE"
--------M-330005-----------------------------
INT 33 - MS MOUSE v1.0+ - RETURN BUTTON PRESS DATA
  AX = 0005h
  BX = button number (see #03169)
```

```
Return: AX = button states (see #03168)
  BX = number of times specified button has been pressed since last call
  CX = column at time specified button was last pressed
  DX = row at time specified button was last pressed
Note: at least for the Genius mouse driver, the number of button presses
    returned is limited to 7FFFh
SeeAlso: AX=0006h,INT 62/AX=007Ch

(Table 03169)
Values for mouse button number:
 0000h  left
 0001h  right
 0002h  middle (Mouse Systems/Logitech/Genius mouse)
--------M-330006-----------------------------
INT 33 - MS MOUSE v1.0+ - RETURN BUTTON RELEASE DATA
  AX = 0006h
  BX = button number (see #03169)
Return: AX = button states (see #03168)
  BX = number of times specified button has been released since last call
  CX = column at time specified button was last released
  DX = row at time specified button was last released
Note: at least for the Genius mouse driver, the number of button releases
    returned is limited to 7FFFh
SeeAlso: AX=0005h,INT 62/AX=007Ch
--------M-330007-----------------------------
INT 33 - MS MOUSE v1.0+ - DEFINE HORIZONTAL CURSOR RANGE
  AX = 0007h
  CX = minimum column
  DX = maximum column
Note: in text modes, the minimum and maximum columns are truncated to the
    next lower multiple of the cell size, typically 8x8 pixels
SeeAlso: AX=0008h,AX=0010h,AX=0031h,INT 62/AX=0080h
SeeAlso: INT 6F/AH=0Ch"F_SET_LIMITS_X"
--------M-330008-----------------------------
INT 33 - MS MOUSE v1.0+ - DEFINE VERTICAL CURSOR RANGE
  AX = 0008h
  CX = minimum row
  DX = maximum row
Note: in text modes, the minimum and maximum rows are truncated to the
    next lower multiple of the cell size, typically 8x8 pixels
SeeAlso: AX=0007h,AX=0010h,AX=0031h,INT 62/AX=0080h
```

```
SeeAlso: INT 6F/AH=0Eh"F_SET_LIMITS_Y"
--------M-330009-----------------------------
INT 33 - MS MOUSE v3.0+ - DEFINE GRAPHICS CURSOR
  AX = 0009h
  BX = column of cursor hot spot in bitmap (-16 to 16)
  CX = row of cursor hot spot (-16 to 16)
  ES:DX -> mask bitmap (see #03170)
Notes:  in graphics modes, the screen contents around the current mouse cursor
    position are ANDed with the screen mask and then XORed with the
    cursor mask
  the Microsoft mouse driver v7.04 and v8.20 uses only BL and CL, so the
    hot spot row/column should be limited to -128..127
  Microsoft KnowledgeBase article Q19850 states that the high bit is
    right-most, but that statement is contradicted by all other available
    documentation
SeeAlso: AX=000Ah,AX=0012h,AX=002Ah,INT 62/AX=007Fh,INT 6F/AH=0Ah"F_DEF_MASKS"

Format of mouse mask bitmap:
Offset  Size  Description (Table 03170)
 00h 16 WORDs screen mask
 10h 16 WORDs cursor mask
Note: each word defines the sixteen pixels of a row, low bit rightmost
--------M-33000A-----------------------------
INT 33 - MS MOUSE v3.0+ - DEFINE TEXT CURSOR
  AX = 000Ah
  BX = hardware/software text cursor
      0000h software
    CX = screen mask
    DX = cursor mask
      0001h hardware
    CX = start scan line
    DX = end scan line
Note: when the software cursor is selected, the character/attribute data at
    the current screen position is ANDed with the screen mask and then
    XORed with the cursor mask
SeeAlso: AX=0009h,INT 62/AX=007Eh
--------M-33000B-----------------------------
INT 33 - MS MOUSE v1.0+ - READ MOTION COUNTERS
  AX = 000Bh
Return: CX = number of mickeys mouse moved horizontally since last call
  DX = number of mickeys mouse moved vertically
```

```
Notes:  a mickey is the smallest increment the mouse can sense
  positive values indicate down/right
SeeAlso: AX=0003h,AX=001Bh,AX=0027h
--------M-33000C-----------------------------
INT 33 - MS MOUSE v1.0+ - DEFINE INTERRUPT SUBROUTINE PARAMETERS
  AX = 000Ch
  CX = call mask (see #03171)
  ES:DX -> FAR routine (see #03172)
SeeAlso: AX=0018h

Bitfields for mouse call mask:
Bit(s)  Description (Table 03171)
 0  call if mouse moves
 1  call if left button pressed
 2  call if left button released
 3  call if right button pressed
 4  call if right button released
 5  call if middle button pressed (Mouse Systems/Logitech/Genius mouse)
 6  call if middle button released (Mouse Systems/Logitech/Genius mouse)
 7-15 unused
Note: some versions of the Microsoft documentation incorrectly state that CX
    bit 0 means call if mouse cursor moves

(Table 03172)
Values interrupt routine is called with:
  AX = condition mask (same bit assignments as call mask)
  BX = button state
  CX = cursor column
  DX = cursor row
  SI = horizontal mickey count
  DI = vertical mickey count
Notes:  some versions of the Microsoft documentation erroneously swap the
    meanings of SI and DI
  in text modes, the row and column will be reported as a multiple of
    the character cell size, typically 8x8 pixels
--------M-33000D-----------------------------
INT 33 - MS MOUSE v1.0+ - LIGHT PEN EMULATION ON
  AX = 000Dh
SeeAlso: AX=000Eh,INT 10/AH=04h
--------M-33000E-----------------------------
INT 33 - MS MOUSE v1.0+ - LIGHT PEN EMULATION OFF
```

```
  AX = 000Eh
SeeAlso: AX=000Dh
--------M-33000F-----------------------------
INT 33 - MS MOUSE v1.0+ - DEFINE MICKEY/PIXEL RATIO
  AX = 000Fh
  CX = number of mickeys per 8 pixels horizontally (default 8)
  DX = number of mickeys per 8 pixels vertically (default 16)
SeeAlso: AX=0013h,AX=001Ah,INT 62/AX=0082h
--------M-330010-----------------------------
INT 33 - MS MOUSE v1.0+ - DEFINE SCREEN REGION FOR UPDATING
  AX = 0010h
  CX,DX = X,Y coordinates of upper left corner
  SI,DI = X,Y coordinates of lower right corner
Note: mouse cursor is hidden in the specified region, and needs to be
    explicitly turned on again
SeeAlso: AX=0001h,AX=0002h,AX=0007h,AX=0010h"Genius MOUSE",AX=0031h
--------M-330010-----------------------------
INT 33 - Genius MOUSE - DEFINE SCREEN REGION FOR UPDATING
  AX = 0010h
  ES:DX -> update region list (see #03173)
Notes:  mouse cursor is hidden in the specified region, and needs to be
    explicitly turned on again
  this version of the call is described in an August 1988 version of the
    Genius Mouse programmer's reference; it has been changed to conform
    to the Microsoft version shown above by version 9.06 (and possibly
    earlier versions)
SeeAlso: AX=0001h,AX=0002h,AX=0007h,AX=0010h"MS MOUSE"

Format of Genius Mouse update region list:
Offset  Size  Description (Table 03173)
 00h  WORD  left-most column
 02h  WORD  top-most row
 04h  WORD  right-most column
 06h  WORD  bottom-most row
--------M-330011-----------------------------
INT 33 - Genius Mouse 9.06 - GET NUMBER OF BUTTONS
  AX = 0011h
Return: AX = FFFFh
  BX = number of buttons
SeeAlso: AX=0000h
--------M-330012-----------------------------
```

```
INT 33 - MS MOUSE - SET LARGE GRAPHICS CURSOR BLOCK
  AX = 0012h
  BH = cursor width in words
  CH = rows in cursor
  BL = horizontal hot spot (-16 to 16)
  CL = vertical hot spot (-16 to 16)
  ES:DX -> bit map of screen and cursor maps
Return: AX = FFFFh if successful
SeeAlso: AX=0009h,AX=002Ah,AX=0035h
--------M-330013-----------------------------
INT 33 - MS MOUSE v5.0+ - DEFINE DOUBLE-SPEED THRESHOLD
  AX = 0013h
  DX = threshold speed in mickeys/second, 0000h = default of 64/second
Note: if speed exceeds threshold, the cursor's on-screen motion is doubled
SeeAlso: AX=000Fh,AX=001Bh,AX=002Ch
--------M-330014-----------------------------
INT 33 - MS MOUSE v3.0+ - EXCHANGE INTERRUPT SUBROUTINES
  AX = 0014h
  CX = call mask (see #03171)
  ES:DX -> FAR routine
Return: CX = call mask of previous interrupt routine
  ES:DX = FAR address of previous interrupt routine
SeeAlso: AX=000Ch,AX=0018h
--------M-330015-----------------------------
INT 33 - MS MOUSE v6.0+ - RETURN DRIVER STORAGE REQUIREMENTS
  AX = 0015h
Return: BX = size of buffer needed to store driver state
SeeAlso: AX=0016h,AX=0017h,AX=0042h
--------M-330016-----------------------------
INT 33 - MS MOUSE v6.0+ - SAVE DRIVER STATE
  AX = 0016h
  BX = size of buffer (see AX=0015h)
  ES:DX -> buffer for driver state
Note: although not documented (since the Microsoft driver does not use it),
    many drivers appear to require BX on input
SeeAlso: AX=0015h,AX=0017h
--------M-330017-----------------------------
INT 33 - MS MOUSE v6.0+ - RESTORE DRIVER STATE
  AX = 0017h
  BX = size of buffer (see AX=0015h)
  ES:DX -> buffer containing saved state
```

```
Notes:  although not documented (since the Microsoft driver does not use it),
    many drivers appear to require BX on input
  some mouse drivers range-check the values in the saved state based on
    the current video mode; thus, the video mode should be restored
    before the mouse driver's state is restored
SeeAlso: AX=0015h,AX=0016h
--------M-330018-----------------------------
INT 33 - MS MOUSE v6.0+ - SET ALTERNATE MOUSE USER HANDLER
  AX = 0018h
  CX = call mask (see #03174)
  ES:DX -> FAR routine to be invoked on mouse events (see #03175)
Return: AX = status
      0018h if successful
      FFFFh on error
Notes:  up to three handlers can be defined by separate calls to this function,
    each with a different combination of shift states in the call mask;
    calling this function again with a call mask of 0000h undefines the
    specified handler (official documentation); specifying the same
    call mask and an address of 0000h:0000h undefines the handler (real
    life)
  some versions of the documentation erroneously reverse the order of
    the bits in the call mask
SeeAlso: AX=000Ch,AX=0014h,AX=0019h

Bitfields for mouse call mask:
Bit(s)  Description (Table 03174)
 0  call if mouse moves
 1  call if left button pressed
 2  call if left button released
 3  call if right button pressed
 4  call if right button released
 5  call if shift button pressed during event
 6  call if ctrl key pressed during event
 7  call if alt key pressed during event
Note: at least one of 5-7 must be set

(Table 03175)
Values user handler is called with:
  AX = condition mask (same bit assignments as call mask)
  BX = button state
  CX = cursor column
```

```
  DX = cursor row
  SI = horizontal mickey count
  DI = vertical mickey count
Return: registers preserved
Note: in text modes, the row and column will be reported as a multiple of
    the cell size, typically 8x8 pixels
--------M-330019-----------------------------
INT 33 - MS MOUSE v6.0+ - RETURN USER ALTERNATE INTERRUPT VECTOR
  AX = 0019h
  CX = call mask (see #03174)
Return: BX:DX = user interrupt vector
  CX = call mask (0000h if not found)
Note: attempts to find a user event handler (defined by function 18h)
    whose call mask matches CX
SeeAlso: AX=0018h
--------M-33001A-----------------------------
INT 33 - MS MOUSE v6.0+ - SET MOUSE SENSITIVITY
  AX = 001Ah
  BX = horizontal speed \
  CX = vertical speed   / (see AX=000Fh)
  DX = double speed threshold (see AX=0013h)
SeeAlso: AX=0013h,AX=001Bh,INT 62/AX=0082h
--------M-33001B-----------------------------
INT 33 - MS MOUSE v6.0+ - RETURN MOUSE SENSITIVITY
  AX = 001Bh
Return: BX = horizontal speed
  CX = vertical speed
  DX = double speed threshold
SeeAlso: AX=000Bh,AX=001Ah
--------M-33001C-----------------------------
INT 33 - MS MOUSE v6.0+ - SET INTERRUPT RATE
  AX = 001Ch
  BX = rate (see #03176)
Notes:  only available on InPort mouse
  values greater than 4 may cause unpredictable driver behavior

(Table 03176)
Values for mouse interrupt rate:
 00h  no interrupts allowed
 01h  30 per second
 02h  50 per second
```

```
 03h  100 per second
 04h  200 per second
--------M-33001D-----------------------------
INT 33 - MS MOUSE v6.0+ - DEFINE DISPLAY PAGE NUMBER
  AX = 001Dh
  BX = display page number
Note: the cursor will be displayed on the specified page
SeeAlso: AX=001Eh
--------M-33001E-----------------------------
INT 33 - MS MOUSE v6.0+ - RETURN DISPLAY PAGE NUMBER
  AX = 001Eh
Return: BX = display page number
SeeAlso: AX=001Dh
--------M-33001F-----------------------------
INT 33 - MS MOUSE v6.0+ - DISABLE MOUSE DRIVER
  AX = 001Fh
Return: AX = status
      001Fh successful
    ES:BX = INT 33 vector before mouse driver was first installed
      FFFFh unsuccessful
Notes:  restores vectors for INT 10 and INT 71 (8086) or INT 74 (286/386)
  if you restore INT 33 to ES:BX, driver will be completely disabled
  many drivers return AX=001Fh even though the driver has been disabled
SeeAlso: AX=0020h
--------M-330020-----------------------------
INT 33 - MS MOUSE v6.0+ - ENABLE MOUSE DRIVER
  AX = 0020h
Return: AX = status
      0020h successful
      FFFFh unsuccessful
Notes:  restores vectors for INT 10h and INT 71h (8086) or INT 74h (286/386)
    which were removed by function 1Fh
  Microsoft's documentation states that no value is returned
SeeAlso: AX=001Fh
--------M-330021-----------------------------
INT 33 - MS MOUSE v6.0+ - SOFTWARE RESET
  AX = 0021h
Return: AX = status
      FFFFh if mouse driver installed
    BX = number of buttons (FFFFh = two buttons)
      0021h if mouse driver not installed
```

```
Note: this call is identical to funtion 00h, but does not reset the mouse
SeeAlso: AX=0000h
--------M-330022-----------------------------
INT 33 - MS MOUSE v6.0+ - SET LANGUAGE FOR MESSAGES
  AX = 0022h
  BX = language (see #03177)
Note: only available on international versions of the driver; US versions
    ignore this call
SeeAlso: AX=0023h

(Table 03177)
Values for mouse driver language:
 00h  English
 01h  French
 02h  Dutch
 03h  German
 04h  Swedish
 05h  Finnish
 06h  Spanish
 07h  Portugese
 08h  Italian
--------M-330023-----------------------------
INT 33 - MS MOUSE v6.0+ - GET LANGUAGE FOR MESSAGES
  AX = 0023h
Return: BX = language (see #03177)
Note: the US version of the driver always returns zero
SeeAlso: AX=0022h
--------M-330024BX0000----------------------
INT 33 - MS MOUSE v6.26+ - GET SOFTWARE VERSION, MOUSE TYPE, AND IRQ NUMBER
  AX = 0024h
  BX = 0000h to check for function's existence
Return: AX = FFFFh on error
  otherwise,
      BH = major version
      BL = minor version
      CH = type (1=bus, 2=serial, 3=InPort, 4=PS/2, 5=HP)
      CL = interrupt (0=PS/2, 2=IRQ2, 3=IRQ3,...,7=IRQ7,...,0Fh=IRQ15)
Note: although current Microsoft documentation states that this function was
    introduced in v6.26, it appears to have been present as early as
    v6.02 (for earlier versions, use INT 33/AX=006Dh)
SeeAlso: AX=004Dh,AX=006Dh
```

```
--------M-330025-----------------------------
INT 33 - MS MOUSE v6.26+ - GET GENERAL DRIVER INFORMATION
  AX = 0025h
Return: AX = general information (see #03178)
  BX = cursor lock flag for OS/2 to prevent reentrancy problems
  CX = mouse code active flag (for OS/2)
  DX = mouse driver busy flag (for OS/2)

Bitfields for general mouse driver information:
Bit(s)  Description (Table 03178)
 15 driver loaded as device driver rather than TSR
 14 driver is newer integrated type
 13,12  current cursor type
  00 software text cursor
  01 hardware text cursor (CRT Controller's cursor)
  1X graphics cursor
 11-8 interrupt rate (see #03176)
 7-0  count of currently-active Mouse Display Drivers (MDD), the newer
    integrated driver type
--------M-330026-----------------------------
INT 33 - MS MOUSE v6.26+ - GET MAXIMUM VIRTUAL COORDINATES
  AX = 0026h
Return: BX = mouse-disabled flag (0000h mouse enabled, nonzero disabled)
  CX = maximum virtual X (for current video mode)
  DX = maximum virtual Y
Note: for driver versions before 7.05, this call returns the currently-set
    maximum coordinates; v7.05+ returns the absolute maximum coordinates
SeeAlso: AX=0031h
--------M-330026-----------------------------
INT 33 - Genius Mouse 9.06 - ???
  AX = 0026h
Return: CX = 0204h if CX was 0105h on entry, else unchanged
--------M-330027-----------------------------
INT 33 - MS MOUSE v7.01+ - GET SCREEN/CURSOR MASKS AND MICKEY COUNTS
  AX = 0027h
Return: AX = screen-mask value (or hardware cursor scan-line start for v7.02+)
  BX = cursor-mask value (or hardware cursor scan-line stop for v7.02+)
  CX = horizontal mickeys moved since last call
  DX = vertical mickeys moved since last call
SeeAlso: AX=000Bh
--------M-330028-----------------------------
```

```
INT 33 - MS MOUSE v7.0+ - SET VIDEO MODE
  AX = 0028h
  CX = new video mode (call is NOP if 0000h)
  DH = Y font size (00h = default)
  DL = X font size (00h = default)
Return: CL = status (00h = successful)
Notes:  DX is ignored unless the selected video mode supports font size control
  when CX=0000h, an internal flag that had been set by a previous call
    is cleared; this is required before a mouse reset
SeeAlso: AX=0029h,INT 10/AH=00h
--------M-330029-----------------------------
INT 33 - MS MOUSE v7.0+ - ENUMERATE VIDEO MODES
  AX = 0029h
  CX = previous video mode
       0000h get first supported video mode
       other get next supported mode after mode CX
Return: CX = first/next video mode (0000h = no more video modes)
  DS:DX -> description of video mode or 0000h:0000h if none
Notes:  the enumerated video modes may be in any order and may repeat
  the description string (if available) is terminated by '$' followed by
    a NUL byte
SeeAlso: AX=0028h
--------M-33002A-----------------------------
INT 33 - MS MOUSE v7.02+ - GET CURSOR HOT SPOT
  AX = 002Ah
Return: AX = internal counter controlling cursor visibility
  BX = cursor hot spot column
  CX = cursor hot spot row
  DX = mouse type (see #03179)
Note: the hot spot location is relative to the upper left corner of the
    cursor block and may range from -128 to +127 both horizontally and
    vertically
SeeAlso: AX=0009h,AX=0012h,AX=0035h

(Table 03179)
Values for mouse type:
 00h  none
 01h  bus
 02h  serial
 03h  InPort
 04h  IBM
```

```
 05h  Hewlett-Packard
--------M-33002B-----------------------------
INT 33 - MS MOUSE v7.0+ - LOAD ACCELERATION PROFILES
  AX = 002Bh
  BX = active acceleration profile
       0001h-0004h or FFFFh to restore default curves
  ES:SI -> buffer containing acceleration profile data (see #03180)
Return: AX = success flag
SeeAlso: AX=002Ch,AX=002Dh,AX=0033h

Format of acceleration profile data:
Offset  Size  Description (Table 03180)
 00h  BYTE  length of acceleration profile 1
 01h  BYTE  length of acceleration profile 2
 02h  BYTE  length of acceleration profile 3
 03h  BYTE  length of acceleration profile 4
 04h 32 BYTEs threshold speeds for acceleration profile 1
 24h 32 BYTEs threshold speeds for acceleration profile 2
 44h 32 BYTEs threshold speeds for acceleration profile 3
 64h 32 BYTEs threshold speeds for acceleration profile 4
 84h 32 BYTEs speedup factor for acceleration profile 1
    (10h = 1.0, 14h = 1.25, 20h = 2.0, etc)
 A4h 32 BYTEs speedup factor for acceleration profile 2
    (10h = 1.0, 14h = 1.25, 20h = 2.0, etc)
 C4h 32 BYTEs speedup factor for acceleration profile 3
    (10h = 1.0, 14h = 1.25, 20h = 2.0, etc)
 E4h 32 BYTEs speedup factor for acceleration profile 4
    (10h = 1.0, 14h = 1.25, 20h = 2.0, etc)
104h 16 BYTEs name of acceleration profile 1 (blank-padded)
114h 16 BYTEs name of acceleration profile 2 (blank-padded)
124h 16 BYTEs name of acceleration profile 3 (blank-padded)
134h 16 BYTEs name of acceleration profile 4 (blank-padded)
Note: unused bytes in the threshold speed fields are filled with 7Fh and
    unused bytes in the speedup factor fields are filled with 10h
--------M-33002C-----------------------------
INT 33 - MS MOUSE v7.0+ - GET ACCELERATION PROFILES
  AX = 002Ch
Return: AX = status (0000h success)
  BX = currently-active acceleration profile
  ES:SI -> acceleration profile data (see #03180)
SeeAlso: AX=002Bh,AX=002Dh,AX=0033h
```

```
--------M-33002D-----------------------------
INT 33 - MS MOUSE v7.0+ - SELECT ACCELERATION PROFILE
  AX = 002Dh
  BX = acceleration level
       0001h-0004h to set profile, or FFFFh to get current profile
Return: AX = status
       0000h successful
     ES:SI -> 16-byte blank-padded name of acceleration profile
       FFFEh invalid acceleration curve number
     ES:SI destroyed
  BX = active acceleration curve number
SeeAlso: AX=0013h,AX=002Bh,AX=002Ch,AX=002Eh
--------M-33002E-----------------------------
INT 33 - MS MOUSE v8.10+ - SET ACCELERATION PROFILE NAMES
  AX = 002Eh
  BL = flag (if nonzero, fill ES:SI buffer with default names on return)
  ES:SI -> 64-byte buffer containing profile names (16 bytes per name)
Return: AX = status (0000h success)
       FFFEh error for ATI Mouse driver
  ES:SI buffer filled with default names if BL nonzero on entry
Notes:  not supported by Logitech driver v6.10
  supported by ATI Mouse driver v7.04
SeeAlso: AX=002Ch,AX=002Dh,AX=012Eh,AX=022Eh
--------M-33002F-----------------------------
INT 33 - MS MOUSE v7.02+ - MOUSE HARDWARE RESET
  AX = 002Fh
Return: AX = status
Note: invoked by mouse driver v8.20 on being called with INT 2F/AX=530Bh
SeeAlso: INT 2F/AH=53h
--------M-330030-----------------------------
INT 33 - MS MOUSE v7.04+ - GET/SET BallPoint INFORMATION
  AX = 0030h
  CX = command
       0000h get status of BallPoint device
       other set rotation angle and masks
     BX = rotation angle (-32768 to 32767 degrees)
     CH = primary button mask
     CL = secondary button mask
Return: AX = button status (FFFFh if no BallPoint) (see #03181)
  BX = rotation angle (0-360 degrees)
  CH = primary button mask
```

```
  CL = secondary button mask
Note: not supported by the ATI Mouse driver which calls itself v7.04

Bitfields for BallPoint mouse button status:
Bit(s)  Description (Table 03181)
 5  button 1
 4  button 2
 3  button 3
 2  button 4
 other  zero
--------M-330031-----------------------------
INT 33 - MS MOUSE v7.05+ - GET CURRENT MINIMUM/MAXIMUM VIRTUAL COORDINATES
  AX = 0031h
Return: AX = virtual X minimum
  BX = virtual Y minimum
  CX = virtual X maximum
  DX = virtual Y maximum
Note: the minimum and maximum values are those set by AX=0007h and AX=0008h;
    the default is minimum = 0 and maximum = absolute maximum
    (see AX=0026h)
SeeAlso: AX=0007h,AX=0008h,AX=0010h,AX=0026h
--------M-330032-----------------------------
INT 33 - MS MOUSE v7.05+ - GET ACTIVE ADVANCED FUNCTIONS
  AX = 0032h
Return: AX = active function flags (FFFFh for v8.10)
      bit 15: function 0025h supported
      bit 14: function 0026h supported
      ...
      bit 0:  function 0034h supported
  BX = ??? (0000h) officially unused
  CX = ??? (E000h) officially unused
  DX = ??? (0000h) officially unused
Note: the Italian version of MS MOUSE v8.20 reportedly indicates that
    functions 0033h and 0034h are not supported even though they are
--------M-330033-----------------------------
INT 33 - MS MOUSE v7.05+ - GET SWITCH SETTINGS AND ACCELERATION PROFILE DATA
  AX = 0033h
  CX = size of buffer
      0000h get required buffer size
    Return: AX = 0000h
      CX = required size (0154h for Logitech v6.10, 0159h
```

```
        for MS v8.10-8.20)
      other
    ES:DX -> buffer of CX bytes for mouse settings
    Return: AX = 0000h
      CX = number of bytes returned
      ES:DX buffer filled (see #03182)
SeeAlso: AX=002Bh

Format of mouse settings data buffer:
Offset  Size  Description (Table 03182)
 00h  BYTE  mouse type
 01h  BYTE  current language
 02h  BYTE  horizontal sensitivity (00h-64h)
 03h  BYTE  vertical sensitivity (00h-64h)
 04h  BYTE  double-speed threshold (00h-64h)
 05h  BYTE  ballistic curve (01h-04h)
 06h  BYTE  interrupt rate (01h-04h)
 07h  BYTE  cursor override mask
 08h  BYTE  laptop adjustment
 09h  BYTE  memory type (00h-02h)
 0Ah  BYTE  SuperVGA support (00h,01h)
 0Bh  BYTE  rotation angle
 0Ch  BYTE  ???
 0Dh  BYTE  primary button (01h-04h)
 0Eh  BYTE  secondary button (01h-04h)
 0Fh  BYTE  click lock enabled (00h,01h)
 10h 324 BYTEs  acceleration profile data (see #03180)
154h  5 BYTEs ??? (Microsoft driver, but not Logitech)
--------M-330034-----------------------------
INT 33 - MS MOUSE v8.0+ - GET INITIALIZATION FILE
  AX = 0034h
Return: AX = status (0000h successful)
  ES:DX -> ASCIZ initialization (.INI) file name
--------M-330035-----------------------------
INT 33 - MS MOUSE v8.10+ - LCD SCREEN LARGE POINTER SUPPORT
  AX = 0035h
  BX = function
      FFFFh get current settings
    Return: AX = 0000h
      BH = style (see #03183)
      BL = size (see #03184)
```

```
      CH = threshold (00h-64h)
      CL = active flag (00h disabled, 01h enabled)
      DX = delay
      other
    BH = pointer style (see #03183)
    BL = size (see #03184)
    CH = threshold (00h-64h)
    CL = active flag (00h disable size change, 01h enable)
    DX = delay (0000h-0064h)
    Return: AX = 0000h
Note: not supported by Logitech driver v6.10
SeeAlso: AX=0012h,AX=002Ah

(Table 03183)
Values for pointer style:
 00h  normal
 01h  reverse
 02h  transparent
SeeAlso: #03184

(Table 03184)
Values for pointer size:
 00h  small ("1")
 01h  medium ("1.5")
 02h  large ("2")
SeeAlso: #03183
--------M-330042-----------------------------
INT 33 - PCMOUSE - GET MSMOUSE STORAGE REQUIREMENTS
  AX = 0042h
Return: AX = status
      0000h MSMOUSE not installed
      0042h functions 42h, 50h, and 52h not supported
      FFFFh successful
    BX = buffer size in bytes for functions 50h and 52h
Note: this function is also supported by the Genius Mouse 9.06 driver
SeeAlso: AX=0015h,AX=0050h,AX=0052h
--------M-330043-----------------------------
INT 33 - Mouse Systems MOUSE DRIVER v7.01+ - CONFIGURE MOUSE???
  AX = 0043h
  CX:BX -> configuration buffer (see #03185)
  DL = ???
```

```
Return: ???
Notes:  also calls routines for INT 33/AX=0053h and INT 33/AX=004Fh
  this function is also supported by the Genius Mouse 9.06 driver

Format of Mouse Systems configuration buffer:
Offset  Size  Description (Table 03185)
 00h  WORD  I/O port address
 02h  BYTE  ???
 03h  BYTE  interrupt number
 04h  BYTE  interrupt mask for interrupt controller
 05h  5 BYTEs ???
--------M-330044CXCDEF-----------------------
INT 33 - Mouse Systems MOUSE DRIVER v7.01+ - TOGGLE IGNORE ACCELERATION CMDS
  AX = 0044h
  CX = CDEFh
Return: AX = new state of "Ignore Application Acceleration Commands" flag
Note: this function is also supported by the Genius Mouse 9.06 driver
SeeAlso: AX=0045h
--------M-330045CXCDEF-----------------------
INT 33 - Mouse Systems MOUSE DRIVER v7.01+ - TOGGLE RESOLUTION DOUBLING
  AX = 0045h
  CX = CDEFh
Return: AX = new state of resolution doubling flag
Note: this function is also supported by the Genius Mouse 9.06 driver
SeeAlso: AX=0044h
--------M-330047-----------------------------
INT 33 - Mouse Systems MOUSE DRIVER v7.01+ - SET BUTTON ASSIGNMENTS
  AX = 0047h
  ES:BX -> button assignments (3 bytes, combinations of "L", "M", "R")
Return: ???
Note: also supported by Genius Mouse 9.06 driver
SeeAlso: AX=0067h
--------M-330048BXCDEF-----------------------
INT 33 - Mouse Systems MOUSE DRIVER v7.01+ - GET ???
  AX = 0048h
  BX = CDEFh
Return: CX = ???
  BH = ???
  BL = ??? (if 50h, driver is using PS/2 pointing device BIOS interface)
Note: also supported by Genius Mouse 9.06 driver
--------M-33004B-----------------------------
```

```
INT 33 - LCS/Telegraphics MOUSE DRIVERS - INSTALLATION CHECK / GET VERSION
  AX = 004Bh
Return: ES:DI -> ASCIZ signature/description string if installed (see #03186)

(Table 03186)
Values for LCS/Telegraphics mouse driver OEM signature/description string:
 "Primax Generic;Universal Mouse Driver;IMOUSE;v8.20i"
 "Synaptics;TouchPad Driver;SYNTOUCH;v2.26"
 "Z-NIX;BUS,AUX,Serial 3-byte and 5-byte Mouse Driver;ZMOUSE;v7.04d"
Note: the string consists of OEM, driver description, driver name, and
    version number
--------M-33004CBXCDEF-----------------------
INT 33 - Mouse Systems MOUSE DRIVER v7.01+ - SET ??? FLAG
  AX = 004Ch
  BX = CDEFh
Note: also supported by Genius Mouse 9.06
SeeAlso: AX=006Ch
--------M-33004D-----------------------------
INT 33 - MS MOUSE - RETURN POINTER TO COPYRIGHT STRING
  AX = 004Dh
Return: ES:DI -> copyright message "*** This is Copyright 1983 Microsoft" or
    "Copyright 19XX...."
Notes:  also supported by Logitech, Kraft, Genius Mouse, and Mouse Systems
    mouse drivers
  in the Genius Mouse 9.06 driver, the ASCIZ signature "KYE" immediately
    follows the above copyright message (KYE Corp. manufactures the
    driver)
SeeAlso: AX=0024h,AX=006Dh,AX=0666h
--------M-33004F-----------------------------
INT 33 - Mouse Systems MOUSE DRIVER v7.01+ - ENABLE MOUSE
  AX = 004Fh
Return: nothing
Note: also supported by Genius Mouse 9.06
SeeAlso: AX=0043h,AX=0053h
--------M-330050-----------------------------
INT 33 - PCMOUSE - SAVE MSMOUSE STATE
  AX = 0050h
  BX = buffer size (ignored by some driver versions)
  ES:DX -> buffer
Return: AX = FFFFh if successful
Notes:  the buffer must be large enough to hold the entire state, or following
```

```
    data will be overwritten by state data in versions which ignore BX;
    use INT 33/AX=0042h to get the required size
  this function is also supported by the Genius Mouse 9.06 driver
SeeAlso: AX=0042h,AX=0052h
--------M-330052-----------------------------
INT 33 - PCMOUSE - RESTORE MSMOUSE STATE
  AX = 0052h
  BX = buffer size (ignored by some driver versions)
  ES:DX -> buffer
Return: AX = FFFFh if successful
Note: also supported by Genius Mouse 9.06 driver
SeeAlso: AX=0050h
--------M-330053-----------------------------
INT 33 - Mouse Systems MOUSE DRIVER v7.01+ - DISABLE MOUSE
  AX = 0053h
Return: nothing
Note: also supported by Genius Mouse 9.06
SeeAlso: AX=0043h,AX=004Fh
--------M-330054CXCDEF-----------------------
INT 33 - Mouse Systems MOUSE DRIVER v7.01+ - SELECT ULTRARES ACCELERATION LEVEL
  AX = 0054h
  CX = CDEFh
  BX = new acceleration level (0-9)
Return: ???
Note: this function is also supported by the Genius Mouse 9.06 driver
SeeAlso: AX=005Ah
--------M-330055-----------------------------
INT 33 - Kraft Mouse - GET ???
  AX = 0055h
Return: CX = ???
  DX = ???
  ES = ???
--------M-330058-----------------------------
INT 33 - Mouse Systems MOUSE DRIVER v7.01+ - ???
  AX = 0058h
Return: AX = CS of driver
  CX:BX = original INT 33 vector
  DX = ???
Note: this function is also supported by the Genius Mouse 9.06 driver
--------M-33005A-----------------------------
INT 33 - Mouse Systems MOUSE DRIVER v7.01+ - SET ULTRARES ACCELERATIONS
```

```
  AX = 005Ah
  CX = number of WORDs to copy (max 0014h, but not range-checked)
  DX:SI -> buffer containing thresholds??? (CX words)
  DX:BX -> buffer containing acceleration values???
    (9*14h words, only first CX of each 14h used)
  ???
Return: CF clear
  ???
Note: this function is also supported by Genius Mouse 9.06
SeeAlso: AX=0054h
--------M-330061BXCDEF-----------------------
INT 33 - Mouse Systems MOUSE DRIVER v7.01+ - ???
  AX = 0061h
  BX = CDEFh
Return: CX = ???
Note: also supported by Genius Mouse 9.06
--------M-330067-----------------------------
INT 33 - Mouse Systems MOUSE DRIVER v7.01+ - GET MOUSE BUTTONS???
  AX = 0067h
Return: BL = number of buttons???
Note: also supported by Genius Mouse 9.06
SeeAlso: AX=0047h
--------M-33006A-----------------------------
INT 33 U - ATI Mouse - INSTALLATION CHECK
  AX = 006Ah
Return: AL = AAh
  AH = ???
  BH = ???
  BL = ???
  CL = ???
  CH = ???
Program: ATI's MOUSE.COM and MOUSE.SYS are drivers for the mouse port found on
    some of ATI's video adapters
SeeAlso: AX=006Dh
--------M-33006C-----------------------------
INT 33 U - TRUEDOX Mouse driver v4.01 - GET/SET HARDWARE PARAMETERS
  AX = 006Ch
  BX = new IRQ (0003h or 0004h), or 0000h to get current values only
  CL = new IRQmask (sent to 8259)
  DX = new base I/O port
Return: BX = current IRQ
```

```
  DX = light pen state???
Note: this is the mouse driver for the Dell Dimension series of computers, by
    TRUEDOX Technology Corporation
SeeAlso: AX=00A1h,AX=0666h
--------M-33006CBXCDEF-----------------------
INT 33 - Mouse Systems MOUSE DRIVER v7.01+ - CLEAR ??? FLAG
  AX = 006Ch
  BX = CDEFh
Note: also supported by Genius Mouse 9.06
SeeAlso: AX=004Ch
--------M-33006D-----------------------------
INT 33 - MS MOUSE - GET VERSION STRING
  AX = 006Dh 'm'
Return: ES:DI -> Microsoft version number of resident driver (see #03187)
Notes:  also supported by Logitech, Mouse Systems, Kraft, and Genius mouse
    drivers
  the Mouse Systems 7.01 and Genius Mouse 9.06 drivers report their
    Microsoft version as 7.00 even though they do not support any of the
    functions from 0025h through 002Dh supported by the MS 7.00 driver
    (the Genius Mouse driver supports function 0026h, but it differs
    from the Microsoft function)
  the TRUEDOX 4.01 driver reports its version as 6.26 through this call,
    but as 6.24 through AX=0024h
  There seems to be no reliable method to distinguish MS MOUSE before
    3.00 from mouse drivers of other vendors.
  Some releases of the MS MOUSE 6.00 erroneously return 6.01 instead of
    their true version number.  In this case, a DI value of 01ABh can
    be used to still detect a 6.00 driver.
  For returned versions 6.02+, INT 33/AX=0024h should be used to retrieve
    more accurate version data.
  True MS MOUSE drivers can also be identified by magic numbers in
    their copyright message, stored in the driver's segment (ES).
    These can be found by scanning the first 2 Kb of the mouse
    driver's segment for a string like: [new since 7.00+]
    "** This is Copyright 1983[-19xx] Microsoft ***" with the
    magic number stored one byte after the signature string.
SeeAlso: AX=0024h,AX=004Dh,AX=006Ah,AX=266Ch

Format of Microsoft version number:
Offset  Size  Description (Table 03187)
 00h  BYTE  major version
```

```
 01h  BYTE  minor version (BCD)

(Table 04087)
Values for Microsoft MOUSE copyright string magic numbers:
 5564h  version 3.00..6.00  (for reported versions up to 5.03, and 6.00)
 557Ch  version 6.01Z..6.24 (for reported versions 6.01..6.24)
 E806h  version 6.25    (for reported version 6.25)
 EB02h  version 6.26..7.04  (for reported version 6.26..7.04)
 0800h  Integrated driver 1.0+  (for reported version 9.x+)
Note: Versions above 7.04 (except for integrated mouse drivers) have a magic
    number representing their version number, e.g. 0507h for version 7.05
--------M-330070BXABCD-----------------------
INT 33 - Mouse Systems MOUSE DRIVER - POPUP.COM - INSTALLATION CHECK
  AX = 0070h
  BX = ABCDh
Return: AX = ABCDh if installed
      BX:CX -> data structure (see #03188)
Notes:  this function is also supported by the Genius Mouse 9.06 driver
  the v7.01 POPUP.COM and menu drivers also check for the signature
    CDh ABh BAh DCh at offset -2Ch from the interrupt handler
  if POPUP is not loaded, the returned data structure contains the proper
    signature at offset 00h, but not at offset 08h

Format of Mouse Systems POPUP.COM data structure:
Offset  Size  Description (Table 03188)
 00h  WORD  signature ABCDh
 02h  DWORD pointer to info structure??? (see #03189)
 06h  2 BYTEs ???
 08h  WORD  signature ABCDh

Format of Mouse Systems POPUP.COM info structure:
Offset  Size  Description (Table 03189)
 00h  WORD  driver version
 02h  8 BYTEs ???
 0Ah  WORD  segment of ???
  ???
--------M-330072BXABCD-----------------------
INT 33 - Mouse Systems MOUSE DRIVER v7.01+ - ???
  AX = 0072h
  BX = ABCDh
Return: ???
```

```
Note: this function is also supported by the Genius Mouse 9.06 driver
--------M-330073BXCDEF-----------------------
INT 33 - Mouse Systems MOUSE DRIVER v7.01+ - GET BUTTON ASSIGNMENTS
  AX = 0073h
  BX = CDEFh
  ES:DX -> 3-byte buffer for button assignments
Return: CX = number of buttons???
  ES:DX buffer filled (default is "LMR")
Note: also supported by Genius Mouse 9.06
SeeAlso: AX=0067h
--------M-3300A0-----------------------------
INT 33 U - TRUEDOX Mouse driver - SET HARDWARE PC MODE (3 button)
  AX = 00A0h
Return: nothing
Note: this function is only available if the mouse mode is switchable
    through the power pins
SeeAlso: AX=006Ch"TRUEDOX",AX=00A1h"TRUEDOX"
--------M-3300A1-----------------------------
INT 33 U - TRUEDOX Mouse driver - SET HARDWARE MS MODE (2 button)
  AX = 00A1h
Return: nothing
Notes:  this function is only available if the mouse mode is switchable
    through the power pins
  this is the mouse driver for the Dell Dimension series of computers, by
    TRUEDOX Technology Corporation
SeeAlso: AX=006Ch"TRUEDOX",AX=00A0h"TRUEDOX",AX=00A6h,AX=0666h
--------M-3300A6-----------------------------
INT 33 U - TRUEDOX Mouse driver - SET RESOLUTION
  AX = 00A6h
  BX = new software resolution
       0001h 50-200 dpi
       0002h 200-400 dpi
       0003h 400-800 dpi
Note: this is the mouse driver for the Dell Dimension series of computers, by
    TRUEDOX Technology Corporation
SeeAlso: AX=00A0h,AX=00A1h,AX=0666h
--------M-3300B0-----------------------------
INT 33 U - LCS/Telegraphics MOUSE DRIVERS - ???
  AX = 00B0h
  ???
Return: ???
```

```
--------M-3300D6-----------------------------
INT 33 - Twiddler TWMOUSE - GET BUTTON/TILT STATE
  AX = 00D6h
Return: AX = current button bitmap
  BX = current X tilt (approximately -200..+200 = -90degrees..+90deg.)
  CX = current Y tilt
Program: the Twiddler is a chording keyboard/mouse combination by Handykey
    Corporation
--------M-3300F0-----------------------------
INT 33 U - LCS/Telegraphics MOUSE DRIVERS - ???
  AX = 00F0h
  ???
Return: ???
--------M-3300F1-----------------------------
INT 33 U - LCS/Telegraphics MOUSE DRIVERS - ???
  AX = 00F1h
  ???
Return: ???
--------M-3300F2-----------------------------
INT 33 U - LCS/Telegraphics MOUSE DRIVERS - ???
  AX = 00F2h
  ???
Return: ???
--------M-3300F3-----------------------------
INT 33 U - LCS/Telegraphics MOUSE DRIVERS - ???
  AX = 00F3h
  ???
Return: ???
--------M-330100CX4752-----------------------
INT 33 - GRTMOUSE v1.00+ - INSTALLATION CHECK
  AX = 0100h
  CX = 4752h ('GR')
  DX = 544Dh ('TM')
Return: AX = 474Dh ('GM') if installed
      CX = version number (CH = major, CL = minor)
Program: GRTMOUSE is a graphical-cursor driver for textmode by Tommer Leyvand
SeeAlso: AX=0101h,AX=0102h,AX=0103h,AX=0104h
--------M-330101-----------------------------
INT 33 - GRTMOUSE v1.00+ - SET MOUSE CURSOR SHAPE
  AX = 0101h
  DS:SI -> 16-byte cursor pattern
```

```
Return: CF clear if successful
SeeAlso: AX=0100h,AX=0102h
--------M-330102-----------------------------
INT 33 - GRTMOUSE v1.00+ - GET MOUSE CURSOR SHAPE
  AX = 0102h
  ES:DI -> 16-byte buffer for cursor pattern
SeeAlso: AX=0100h,AX=0101h
--------M-330103-----------------------------
INT 33 - GRTMOUSE v1.00+ - SET ACTIVE CHARACTERS
  AX = 0103h
  CH,CL,DH,DL = ASCII codes to be remapped to display mouse pointer
Note: the default active characters are D0h,D1h,D6h,D8h; the active
   characters should be in the range C0h to DFh
SeeAlso: AX=0100h,AX=0104h
--------M-330104-----------------------------
INT 33 - GRTMOUSE v1.00+ - GET ACTIVE CHARACTERS
  AX = 0104h
Return: CH,CL,DH,DL = ASCII codes for the active characters
SeeAlso: AX=0100h,AX=0103h
--------M-33012E-----------------------------
INT 33 - MS MOUSE v8.10+ - ???
  AX = 012Eh
  BL = ???
Return: AX = 0000h (MS)
  AX = FFFFh (ATI Mouse v7.04)
Note: not supported by Logitech driver v6.10
SeeAlso: AX=002Eh,AX=022Eh
--------M-33022E-----------------------------
INT 33 - MS MOUSE v8.10+ - ???
  AX = 022Eh
  BL = ???
Return: AX = 0000h (MS)
  AX = FFFFh (ATI Mouse v7.04)
Note: not supported by Logitech driver v6.10
SeeAlso: AX=002Eh,AX=012Eh
--------M-330666-----------------------------
INT 33 U - TRUEDOX Mouse driver v4.01 - GET COPYRIGHT STRING
  AX = 0666h
Return: DX:AX -> ASCII "Copyright 1987-1992 TRUEDOX Technology Corporation"
Note: this is the mouse driver for the Dell Dimension series of computers,
    by TRUEDOX Technology Corporation
```

```
SeeAlso: AX=004Dh,AX=00A6h,AX=0666h
--------M-33136C-----------------------------
INT 33 - LOGITECH MOUSE v6.10+ - ???
  AX = 136Ch
  BX = ???
Return: AX = ???
  BX = ???
--------M-33146C-----------------------------
INT 33 - LOGITECH MOUSE v6.10+ - GET/SET ???
  AX = 146Ch
  BL = function
       00h set ???
    BH = new value (zero/nonzero to clear/set)
      else get ???
    Return: ???
--------M-33156C-----------------------------
INT 33 - LOGITECH MOUSE v6.10+ - GET SIGNATURE AND VERSION STRINGS
  AX = 156Ch
Return: ES:DI -> signature "LOGITECH MOUSE DRIVER"
  ES:SI -> version string, terminated with CRLF
--------M-33166C-----------------------------
INT 33 - LOGITECH MOUSE v6.10+ - ???
  AX = 166Ch
  BL = ???
       00h ???
       01h ???
       other ???
    BH = new value of ???
    Return: AX = FFFFh
--------M-33176C-----------------------------
INT 33 - LOGITECH MOUSE v6.10+ - ???
  AX = 176Ch
  ???
Return: ???
--------M-33186C-----------------------------
INT 33 - LOGITECH MOUSE v6.10+ - ???
  AX = 186Ch
  ???
Return: ???
--------M-33196C-----------------------------
INT 33 - LOGITECH MOUSE v6.10+ - ???
```

```
  AX = 196Ch
  ???
Return: ???
--------M-331A6C-----------------------------
INT 33 - LOGITECH MOUSE v6.10+ - GET ???
  AX = 1A6Ch
Return: AX = FFFFh
  BX = ???
  CX = ???
SeeAlso: AX=1B6Ch
--------M-331B6C-----------------------------
INT 33 - LOGITECH MOUSE v6.10+ - SET ???
  AX = 1B6Ch
  BX = new value for ??? (0000h-0003h)
Return: AX = FFFFh
SeeAlso: AX=1A6Ch
--------M-331C6C-----------------------------
INT 33 - LOGITECH MOUSE v6.10+ - ???
  AX = 1C6Ch
  BX = ???
      <42h ???
      =42h ???
      >42h ???
    ES:DI -> ???
    Return: AX = ???
--------M-331D6C-----------------------------
INT 33 - LOGITECH MOUSE - GET COMPASS PARAMETER
  AX = 1D6Ch
Return: BX = direction (0=north, 1=south, 2=east, 3=west)
SeeAlso: AX=1E6Ch
--------M-331E6C-----------------------------
INT 33 - LOGITECH MOUSE - SET COMPASS PARAMETER
  AX = 1E6Ch
  BX = direction (0=north, 1=south, 2=east, 3=west)
SeeAlso: AX=1D6Ch
--------M-331F6C-----------------------------
INT 33 - LOGITECH MOUSE - GET BALLISTICS INFORMATION
  AX = 1F6Ch
Return: BX = 0=off, 1=on
  CX = 1=low, 2=high
SeeAlso: AX=002Ch,AX=236Ch
```

```
--------M-33206C-----------------------------
INT 33 - LOGITECH MOUSE - SET LEFT OR RIGHT PARAMETER
  AX = 206Ch
  BX = parameter (00h = right, FFh = left)
SeeAlso: AX=216Ch
--------M-33216C-----------------------------
INT 33 - LOGITECH MOUSE - GET LEFT OR RIGHT PARAMETER
  AX = 216Ch
Return: BX = parameter (00h = right, FFh = left)
SeeAlso: AX=206Ch
--------M-33226C-----------------------------
INT 33 - LOGITECH MOUSE - REMOVE DRIVER FROM MEMORY
  AX = 226Ch
Note: this only frees memory; does not restore hooked interrupts
--------M-33236C-----------------------------
INT 33 - LOGITECH MOUSE - SET BALLISTICS INFORMATION
  AX = 236Ch
  BX = 0=off, 1=on
  CX = 1=low, 2=high
SeeAlso: AX=002Ch,AX=1F6Ch
--------M-33246C-----------------------------
INT 33 - LOGITECH MOUSE - GET PARAMETERS AND RESET SERIAL MOUSE
  AX = 246Ch
  ES:DX -> parameter table buffer (see #03190)
Return: AX = FFFFh if driver installed for serial mouse
SeeAlso: AX=0000h,AX=256Ch/BX=0000h,AX=256Ch/BX=0001h,AX=256Ch/BX=0003h

Format of Logitech Mouse parameter table:
Offset  Size  Description (Table 03190)
 00h  WORD  baud rate divided by 100  (serial mouse only)
 02h  WORD  emulation     (serial mouse only)
 04h  WORD  report rate     (serial mouse only)
 06h  WORD  firmware revision   (serial mouse only)
 08h  WORD  00h       (serial mouse only)
 0Ah  WORD  port        (serial mouse only)
 0Ch  WORD  physical buttons
 0Eh  WORD  logical buttons
--------M-33256CBX0000----------------------
INT 33 - LOGITECH MOUSE - SET PARAMETERS - SET BAUD RATE (SERIAL MOUSE ONLY)
  AX = 256Ch
  BX = 0000h
```

```
  CX = rate (0=1200, 1=2400, 2=4800, 3=9600)
Return: AX = FFFFh if driver installed for serial mouse
SeeAlso: AX=246Ch,AX=256Ch/BX=0001h,AX=256Ch/BX=0002h,AX=276Ch
--------M-33256CBX0001-----------------------
INT 33 - LOGITECH MOUSE - SET PARAMETERS - SET EMULATION (SERIAL MOUSE ONLY)
  AX = 256Ch
  BX = 0001h
  CX = emulation type (see #03191)
Return: AX = FFFFh if driver installed for serial mouse
SeeAlso: AX=246Ch,AX=256Ch/BX=0000h,AX=256Ch/BX=0003h,AX=276Ch

(Table 03191)
Values for Logitech mouse emulation type:
 00h  5 byte packed binary
 01h  3 byte packed binary
 02h  hexadecimal
 03h  relative bit pad
 04h  not supported
 05h  MM Series
 06h  not supported
 07h  Microsoft
--------M-33256CBX0002-----------------------
INT 33 - LOGITECH MOUSE - SET PARAMETERS - SET REPORT RATE (SERIAL MOUSE ONLY)
  AX = 256Ch
  BX = 0002h
  CX = rate (0=10, 1=20, 2=35, 3=50, 4=70, 5=100, 6=150)
Return: AX = FFFFh if driver installed for serial mouse
SeeAlso: AX=246Ch,AX=256Ch/BX=0001h,AX=256Ch/BX=0003h,AX=276Ch
--------M-33256CBX0003-----------------------
INT 33 - LOGITECH MOUSE - SET PARAMETERS - SET MOUSE PORT (SERIAL MOUSE ONLY)
  AX = 256Ch
  BX = 0003h
  CX = port (1, 2)
Return: AX = FFFFh if driver installed for serial mouse
SeeAlso: AX=246Ch,AX=256Ch/BX=0000h,AX=256Ch/BX=0004h,AX=276Ch
--------M-33256CBX0004-----------------------
INT 33 - LOGITECH MOUSE - SET PARAMETERS - SET MOUSE LOGICAL BUTTONS
  AX = 256Ch
  BX = 0004h
  CX = buttons (2, 3)
Return: AX = FFFFh if driver installed for serial mouse
```

```
SeeAlso: AX=246Ch,AX=276Ch
--------M-33266C-----------------------------
INT 33 - LOGITECH MOUSE - GET VERSION???
  AX = 266Ch
Return: BX = 'SS'
  CH = '4'  major version number
  CL = '1'  minor version number
SeeAlso: AX=006Dh
--------M-33276C-----------------------------
INT 33 - LOGITECH MOUSE - ??? Tries MMSeries, Baud 2400
  AX = 276Ch
SeeAlso: AX=256Ch
--------M-333000-----------------------------
INT 33 - Smooth Mouse Driver, PrecisePoint - INSTALLATION CHECK
  AX = 3000h
Return: AX = FFFFh if installed
      BX = version number (BH = major, BL = minor)
Program: SMD is a programmer's library by Andy Hakim which provides a
    graphics-style mouse cursor in text mode.  PrecisePoint is an
    SMD-based TSR which replaces the block mouse cursor in text
    applications.
SeeAlso: AX=0000h,AX=3001h,AX=3003h
--------M-333001-----------------------------
INT 33 - Smooth Mouse Driver, PrecisePoint - ENABLE SMOOTH MOUSE
  AX = 3001h
Return: AX = status (0000h = disabled, 0001h = enabled)
Note: SMD remains disabled if running under Desqview or in graphics mode
SeeAlso: AX=0001h,AX=0002h,AX=3002h
--------M-333002-----------------------------
INT 33 - Smooth Mouse Driver, PrecisePoint - DISABLE SMOOTH MOUSE
  AX = 3002h
Return: AX = status (0000h = disabled, 0001h = enabled)
SeeAlso: AX=0001h,AX=0002h,AX=3000h,AX=3001h
--------M-333003-----------------------------
INT 33 - Smooth Mouse Driver, PrecisePoint - GET INFORMATION
  AX = 3003h
  BL = data structure selector
       00h Primary Bitmap (used for 25 line mode) (see #03192)
       01h Secondary Bitmap (used for 43/50 line modes) (see #03192)
       02h Sacrifice Character Map (see #03193)
       03h Program Information (see #03194)
```

```
Return: ES:DX -> selected data structure
SeeAlso: AX=3000h

Format of Primary/Secondary Bitmap [SMD_BITMAP_STRUCT]:
Offset  Size  Description (Table 03192)
 00h  BYTE  vertical size of bitmap (00h - 10h)
 01h  BYTE  horizontal size of bitmap (00h - 10h)
 02h  BYTE  vertical hotspot position (00h - 10h)
 03h  BYTE  horizontal hotspot position (00h - 10h)
 04h 16 WORDs cursor bitmap data
 14h 16 WORDs screen bitmap data

Format of Sacrifice Character Map [SMD_SMAP_STRUCT]:
Offset  Size  Description (Table 03193)
 00h  BYTE  bytes are character values (00h-FFh) used in place of the
 01h  BYTE  actual character for the corresponding position on the screen
 02h  BYTE       +--------------+   occupied by part or all of the mouse
 03h  BYTE       | 0h | 1h | 2h |   cursor
 04h  BYTE       |----+----+----|
 05h  BYTE       | 3h | 4h | 5h |
 06h  BYTE       |----+----+----|
 07h  BYTE       | 6h | 7h | 8h |
 08h  BYTE       +--------------+

Format of Program Information [SMD_INFO_STRUCT]:
Offset  Size  Description (Table 03194)
 00h  WORD  segment of old interrupt 33h handler
 02h  WORD  offset of old interrupt 33h handler
 04h  WORD  PSP of SMD
 06h  BYTE  ENABLE/DISABLE manual setting status
 07h  BYTE  ENABLE/DISABLE internal usage status
--------M-333004-----------------------------
INT 33 - Smooth Mouse Driver, PrecisePoint - RESERVED FUTURE EXPANSION
  AX = 3004h
SeeAlso: AX=3000h
--------M-333005-----------------------------
INT 33 - Smooth Mouse Driver, PrecisePoint - RESERVED FUTURE EXPANSION
  AX = 3005h
SeeAlso: AX=3000h
--------M-334F00-----------------------------
INT 33 - LOGITECH MOUSE v6.10+ - GET ???
```

```
  AX = 4F00h
Return: AX = 004Fh if supported
  BX = ???
  ES:DI -> ???
SeeAlso: AX=4F01h
--------M-334F01----------------------------
INT 33 - LOGITECH MOUSE v6.10+ - ???
  AX = 4F01h
  ES = ???
Return: AX = 004Fh if supported
  ES:DI -> ???
SeeAlso: AX=4F00h
--------M-336F00----------------------------
INT 33 - Hewlett Packard - HP MOUSE DRIVER INSTALLATION CHECK
  AX = 6F00h
  BX <> 4850h
Return: BX = 4850h ('HP') if mouse driver written by Hewlett Packard
SeeAlso: INT 10/AX=6F00h,INT 14/AX=6F00h,INT 16/AX=6F00h,INT 17/AX=6F00h
--------M-338800----------------------------
INT 33 U - InfoTrack IMOUSE.COM - UNHOOK MOUSE IRQ
  AX = 8800h
  BX <> FFFFh
Note: the code is written to expect a subfunction number in AL, but only
    function 00h has been implemented
SeeAlso: AX=8800h/BX=FFFFh
--------M-338800BXFFFF----------------------
INT 33 U - InfoTrack IMOUSE.COM - GET ACTIVE IRQ
  AX = 8800h
  BX = FFFFh
Return: BL = number of IRQ being used by the mouse
SeeAlso: AX=8800h
--------T-33FFE6----------------------------
INT 33 - Switch-It v3.23 - GET ??? PROGRAM
  AX = FFE6h
  CX = length of buffer
  ES:DI -> buffer for program name
Return: ES:DI buffer filled
Program: Switch-It is a task switcher supporting up to 100 programs
    simultaneously by Better Software Technology, Inc.
--------T-33FFE7----------------------------
INT 33 - Switch-It v3.23 - GET ???
```

```
  AX = FFE7h
Return: AX = ???
--------T-33FFE8----------------------------
INT 33 - Switch-It v3.23 - ???
  AX = FFE8h
  CX = length of name including terminating NUL
  DS:SI -> ASCIZ program pathname
--------T-33FFE9----------------------------
INT 33 - Switch-It v3.23 - SET ???
  AX = FFE9h
  BX = ???
--------T-33FFEA----------------------------
INT 33 - Switch-It v3.23 - SET ???
  AX = FFEAh
  BL = ???
--------T-33FFEB----------------------------
INT 33 - Switch-It v3.23 - SET ??? FLAG
  AX = FFEBh
--------T-33FFEC----------------------------
INT 33 - Switch-It v3.23 - SET ???
  AX = FFECh
  BL = ???
--------T-33FFED----------------------------
INT 33 - Switch-It v3.23 - GET ???
  AX = FFEDh
Return: AX = ??? (0001h)
  BX = ???
Program: Switch-It is a task switcher supporting up to 100 programs
    simultaneously by Better Software Technology, Inc.
--------T-33FFEE----------------------------
INT 33 - Switch-It v3.23 - GET ???
  AX = FFEEh
Return: AX = ???
--------T-33FFEF----------------------------
INT 33 - Switch-It v3.23 - GET ???
  AX = FFEFh
Return: BX:AX -> ???
--------T-33FFF0----------------------------
INT 33 - Switch-It v3.23 - SET ???
  AX = FFF0h
  BL = ???
```

```
--------T-33FFF1-----------------------------
INT 33 - Switch-It v3.23 - GET CONFIGURATION FILE
  AX = FFF1h
Return: BX:AX -> ASCIZ pathname of configuration file
Program: Switch-It is a task switcher supporting up to 100 programs
    simultaneously by Better Software Technology, Inc.
--------T-33FFF2-----------------------------
INT 33 - Switch-It v3.23 - SET ??? FLAG
  AX = FFF2h
Return: AL = 01h
--------T-33FFF3-----------------------------
INT 33 - Switch-It v3.23 - GET ???
  AX = FFF3h
Return: AX = ???
--------T-33FFF4-----------------------------
INT 33 - Switch-It v3.23 - SET ???
  AX = FFF4h
  BX = ???
  CX = ???
--------T-33FFF5-----------------------------
INT 33 - Switch-It v3.23 - GET ???
  AX = FFF5h
Return: AX = ???
--------T-33FFF6-----------------------------
INT 33 - Switch-It v3.23 - GET ???
  AX = FFF6h
Return: AX = ???
--------T-33FFF7-----------------------------
INT 33 - Switch-It v3.23 - GET ???
  AX = FFF7h
  BX = index of ???
Return: AX = ???
--------T-33FFF8-----------------------------
INT 33 - Switch-It v3.23 - ???
  AX = FFF8h
  BX = ???
  CX = length of program name, including terminating NUL
  DS:SI -> ASCIZ program pathname
Return: ???
Program: Switch-It is a task switcher supporting up to 100 programs
    simultaneously by Better Software Technology, Inc.
```

```
--------T-33FFF9----------------------------
INT 33 - Switch-It v3.23 - NOP
  AX = FFF9h
--------T-33FFFA----------------------------
INT 33 - Switch-It v3.23 - SET ???
  AX = FFFAh
  BX = index of program
SeeAlso: AX=FFFBh,AX=FFFCh
--------T-33FFFB----------------------------
INT 33 - Switch-It v3.23 - GET ???
  AX = FFFBh
  BX = index of program
Return: AX = ??? (0000h or 0001h)
SeeAlso: AX=FFFAh,AX=FFFCh
--------T-33FFFC----------------------------
INT 33 - Switch-It v3.23 - CLEAR ???
  AX = FFFCh
  BX = index of program
SeeAlso: AX=FFFAh,AX=FFFCh
--------T-33FFFD----------------------------
INT 33 - Switch-It v3.23 - GET MEMORY ADDRESSES???
  AX = FFFDh
Return: AX = first available segment???
  BX = paragraph of top of conventional memory
  DX = PSP segment of SI.EXE
--------T-33FFFE----------------------------
INT 33 - Switch-It v3.23 - INSTALLATION CHECK
  AX = FFFEh
Return: BX = ???
  DX = 5349h ("SI")
--------T-33FFFF----------------------------
INT 33 - Switch-It v3.23 - ???
  AX = FFFFh
  BX = ???
Program: Switch-It is a task switcher supporting up to 100 programs
    simultaneously by Better Software Technology, Inc.
--------r-34--------------------------------
INT 34 - FLOATING POINT EMULATION - OPCODE D8h
Desc: this interrupt is used to emulate floating-point instructions with
    an opcode of D8h
Note: the floating-point emulators in Borland and Microsoft languages and
```

```
    Lahey FORTRAN use this interrupt
SeeAlso: INT 35,INT 3E
--------r-35---------------------------------
INT 35 - FLOATING POINT EMULATION - OPCODE D9h
Desc: this interrupt is used to emulate floating-point instructions with
    an opcode of D9h
Note: the floating-point emulators in Borland and Microsoft languages and
    Lahey FORTRAN use this interrupt
SeeAlso: INT 34,INT 36
--------r-36---------------------------------
INT 36 - FLOATING POINT EMULATION - OPCODE DAh
Desc: this interrupt is used to emulate floating-point instructions with
    an opcode of DAh
Note: the floating-point emulators in Borland and Microsoft languages and
    Lahey FORTRAN use this interrupt
SeeAlso: INT 35,INT 37
--------r-37---------------------------------
INT 37 - FLOATING POINT EMULATION - OPCODE DBh
Desc: this interrupt is used to emulate floating-point instructions with
    an opcode of DBh
Note: the floating-point emulators in Borland and Microsoft languages and
    Lahey FORTRAN use this interrupt
SeeAlso: INT 36,INT 38
--------r-38---------------------------------
INT 38 - FLOATING POINT EMULATION - OPCODE DCh
Desc: this interrupt is used to emulate floating-point instructions with
    an opcode of DCh
Note: the floating-point emulators in Borland and Microsoft languages and
    Lahey FORTRAN use this interrupt
SeeAlso: INT 37,INT 39
--------O-38---------------------------------
INT 38 - PC-MOS/386 v3.0 - API
Note: this API was been moved to INT D4h sometime between versions 3.0 and
    5.01; v3.0 supported at least functions 02h,04h,0703h,10h,11h, and
    12h
SeeAlso: INT D4/AH=02h,INT D4/AH=04h,INT D4/AH=07h,INT D4/AH=10h,INT D4/AH=11h
--------r-39---------------------------------
INT 39 - FLOATING POINT EMULATION - OPCODE DDh
Desc: this interrupt is used to emulate floating-point instructions with
    an opcode of DDh
Note: the floating-point emulators in Borland and Microsoft languages and
```

```
    Lahey FORTRAN use this interrupt
SeeAlso: INT 38,INT 3A
--------r-3A---------------------------------
INT 3A - FLOATING POINT EMULATION - OPCODE DEh
Desc: this interrupt is used to emulate floating-point instructions with
    an opcode of DEh
Note: the floating-point emulators in Borland and Microsoft languages and
    Lahey FORTRAN use this interrupt
SeeAlso: INT 39,INT 3B
--------r-3B---------------------------------
INT 3B - FLOATING POINT EMULATION - OPCODE DFh
Desc: this interrupt is used to emulate floating-point instructions with
    an opcode of DFh
Note: the floating-point emulators in Borland and Microsoft languages and
    Lahey FORTRAN use this interrupt
SeeAlso: INT 3A,INT 3C
--------r-3C---------------------------------
INT 3C - FLOATING POINT EMULATION - INSTRUCTIONS WITH SEGMENT OVERRIDE
Notes:  the floating-point emulators in Borland and Microsoft languages and
    Lahey FORTRAN use this interrupt
  the generated code is  CD 3C xy mm ....
    where xy is a modified ESC instruction and mm is the modR/M byte.
    The xy byte appears to be encoded as
    s s 0 1 1 x x x   or  s s 0 0 0 x x x
    where "ss" specifies the segment override:
    00 -> DS:
    01 -> SS:
    10 -> CS:
    11 -> ES:
SeeAlso: INT 3B,INT 3D
--------r-3D---------------------------------
INT 3D - FLOATING POINT EMULATION - STANDALONE FWAIT
Notes:  the floating-point emulators in Borland and Microsoft languages and
    Lahey FORTRAN use this interrupt
  this vector is modified but not restored by Direct Access v4.0, and
    may be left dangling by other programs written with the same version
    of compiled BASIC
SeeAlso: INT 3C,INT 3E
--------r-3E---------------------------------
INT 3E - FLOATING POINT EMULATION - Borland LANGUAGES "SHORTCUT" CALL
Notes:  the two bytes following the INT 3E instruction are the subcode
```

```
    (see #03195) and a NOP (90h), except for subcodes DCh and DEh, where
    the second byte is a register count (01h-08h)
  this vector is modified but not restored by Direct Access v4.0, and
    may be left dangling by other programs written with the same version
    of compiled BASIC
SeeAlso: INT 3D

(Table 03195)
Values for Borland floating-point shortcut subcode:
Subcode   Function
 DCh  load 8086 stack with 8087 registers; overwrites the 10*N bytes at the
    top of the stack prior to the INT 3E with the 8087 register contents
 DEh  load 8087 registers from top of 8086 stack; ST0 is furthest from top
    of 8086 stack
 E0h  round TOS and R1 to single precision, compare, pop twice
    returns AX=8087 status word, FLAGS=8087 condition bits
 E2h  round TOS and R1 to double precision, compare, pop twice
    returns AX=8087 status word, FLAGS=8087 condition bits
  Note: buggy in TPas5.5, because it sets the 8087 precision control
    field to the undocumented value 01h; this results in actually
    rounding to single precision
 E4h  compare TOS/R1 with two POP's
    returns FLAGS=8087 condition bits
 E6h  compare TOS/R1 with POP
    returns FLAGS=8087 condition bits
 E8h  FTST (check TOS value)
    returns FLAGS=8087 condition bits
 EAh  FXAM (check TOS value)
    returns AX=8087 status word
 ECh  sine(ST0)
 EEh  cosine(ST0)
 F0h  tangent(ST0)
 F2h  arctangent(ST0)
 F4h  ST0 = ln(ST0)
 F6h  ST0 = log2(ST0)
 F8h  ST0 = log10(ST0)
 FAh  ST0 = e**ST0
 FCh  ST0 = 2**ST0
 FEh  ST0 = 10**ST0
--------r-3F---------------------------------
INT 3F - Overlay manager interrupt (Microsoft LINK.EXE, Borland TLINK VROOMM)
```

```
Notes:  INT 3F is the default, and may be overridden while linking
  this vector is modified but not restored by Direct Access v4.0, and
    may be left dangling by other programs written with the same version
    of compiled BASIC
SeeAlso: INT FE"OVERLAY"
--------r-3F--------------------------------
INT 3F - Microsoft Dynamic Link Library manager
SeeAlso: INT 21/AH=4Bh
--------B-40--------------------------------
INT 40 - DISKETTE - ROM BIOS DISKETTE HANDLER RELOCATED BY HARD DISK BIOS
SeeAlso: INT 13/AH=00h,INT 13/AH=02h,INT 47"SuperBIOS",INT 63"Adaptec"
--------h-40--------------------------------
INT 40 - Z100 - Master 8259 - Parity error or S100 error
SeeAlso: INT 41"Z100",INT FF"Z100"
--------O-40--------------------------------
INT 40 - Acorn BBC Master 512 - "OSFIND" - OPEN FILE
  AL = operation
       00h close file
       40h open file for reading
       80h open file for writing
       C0h open file for random access
  DS:BX -> CR-terminated filename
Return: AL = file handle (00h if file closed or could not be opened)
Note: the Acorn BBC Master 512 is an 80186-based add-on board for the
    6502-based Master 128 which uses the original CPU as an I/O processor
SeeAlso: INT 41"Acorn",INT 42"Acorn",INT 43"Acorn",INT 44"Acorn",INT 4C"Acorn"
--------h-40--------------------------------
INT 40 - TI Professional PC - IRQ0
Note: on the TI Pro, IRQ0 is connected to the same pin on the expansion bus
    that IBM connects to IRQ2
SeeAlso: INT 0A"IRQ2",INT 41"TI Professional"
--------B-41--------------------------------
INT 41 - SYSTEM DATA - HARD DISK 0 PARAMETER TABLE ADDRESS [NOT A VECTOR!]
Notes:  the default parameter table array is located at F000h:E401h in 100%
    compatible BIOSes; the pointer may be overridden by the hard disk
    controller's BIOS to support drive formats unknown to the ROM BIOS
  not used by some PS/2 models
  BIOSes which support four hard drives may store the parameter tables
    for drives 81h-83h immediately following the parameter table pointed
    at by INT 41, with a separate copy of the drive 81h table for INT 46.
    The check for such an arrangement is to test whether INT 46 points
```

```
    somewhere other than exactly 16 bytes past INT 41, and the sixteen
    bytes starting at offset 10h from INT 41 are identical to the sixteen
    bytes pointed at by INT 46
  another arrangement for BIOSes which support four IDE drives is to have
    four tables pointed at by INT 41 in the order primary master,
    primary slave, secondary master, and secondary slave, in which case
    (for example) a system with only primary master and secondary master
    will have valid tables at offsets 00h and 20h, with garbage (but
    sectors-per-track = 00h) at offsets 10h and 30h
SeeAlso: #03196,INT 13/AH=09h,INT 1E,INT 46"HARD DISK 1",INT 60"Adaptec"
SeeAlso: INT C0"AMI"

Format of fixed disk parameters:
Offset  Size  Description (Table 03196)
 00h  WORD  number of cylinders
 02h  BYTE  number of heads
 03h  WORD  starting reduced write current cylinder (XT only, 0 for others)
 05h  WORD  starting write precompensation cylinder number
 07h  BYTE  maximum ECC burst length (XT only)
 08h  BYTE  control byte (see #03197,#03198)
 09h  BYTE  standard timeout (XT only, 0 for others)
 0Ah  BYTE  formatting timeout (XT and WD1002 only, 0 for others)
 0Bh  BYTE  timeout for checking drive (XT and WD1002 only, 0 for others)
 0Ch  WORD  cylinder number of landing zone (AT and later only)
 0Eh  BYTE  number of sectors per track (AT and later only)
 0Fh  BYTE  reserved
SeeAlso: #00273,#00277

Bitfields for XT fixed disk control byte:
Bit(s)  Description (Table 03197)
 2-0  drive step speed
  000  3ms
  100  200ms
  101  70ms (default)
  110  3ms
  111  3ms
 5-3  unused
 6  disable ECC retries
 7  disable access retries

Bitfields for AT fixed disk control byte:
```

```
Bit(s)  Description (Table 03198)
 0  unused
 1  reserved (0)  (disable IRQ)
 2  reserved (0)  (no reset)
 3  set if more than 8 heads
 4  always 0
 5  set if manufacturer's defect map on max cylinder+1  (AT and later only)
 6  disable ECC retries
 7  disable access retries
--------h-41---------------------------------
INT 41 - Z100 - Master 8259 - Processor Swap
SeeAlso: INT 40"Z100",INT 42"Z100"
--------h-41---------------------------------
INT 41 - TI Professional PC - IRQ1
Note: on the TI Pro, IRQ1 is connected to the same pin on the expansion bus
    that IBM connects to IRQ3
SeeAlso: INT 0B"IRQ3",INT 40"TI Professional",INT 42"TI Professional"
--------O-41---------------------------------
INT 41 - Acorn BBC Master 512 - "OSGBPB" - MULTI-BYTE GET/PUT
  AL = function
       01h put bytes sequentially
       02h put bytes, ignoring sequential pointer
       03h get bytes sequentially
       04h get bytes, ignoring sequential pointer
       05h get media title and boot option
       06h get current device and directory
       07h get current library and device
       08h search directory
  DS:BX -> control block (see #03199)
Return: CF clear if successful
  CF set on error
  AL = 00h if operation attempted
  AL unchanged if unsupported function
SeeAlso: INT 40"Acorn",INT 42"Acorn",INT 43"Acorn"

Format of BBC Master control block:
Offset  Size  Description (Table 03199)
 00h  BYTE  file handle
 01h  DWORD pointer to data in either I/O processor or Tube processor
 05h  DWORD number of bytes to be transferred
 09h  DWORD transfer address
```

```
--------G-410000-----------------------------
INT 41 CPU - MS Windows debugging kernel - OUTPUT CHARACTER FOR USER
  AX = 0000h
  DS:DX -> character
Note: the kernel calls this function when it wants the user program to
    output a character
SeeAlso: AX=0001h
--------G-410001-----------------------------
INT 41 CPU - MS Windows debugging kernel - INPUT CHARACTER
  AX = 0001h
Return: AL = character
Note: the kernel calls this function when it needs to input a character
SeeAlso: AX=0000h
--------G-41000D-----------------------------
INT 41 CPU - MS Windows debugging kernel - TASK GOING OUT
  AX = 000Dh
SeeAlso: AX=000Eh
--------G-41000E-----------------------------
INT 41 CPU - MS Windows debugging kernel - TASK COMING IN
  AX = 000Eh
SeeAlso: AX=000Dh
--------G-410012-----------------------------
INT 41 CPU - MS Windows debugging kernel - "OutputDebugString"
  AX = 0012h
  DS:SI -> string (Windows 3.0)
  ES:SI -> string (Windows 3.1)
Return: nothing???
Note: this function is called by the kernel when it wants to output a
    string through the debugger
SeeAlso: AX=0050h,INT 68/AH=47h
--------G-41004F-----------------------------
INT 41 CPU - MS Windows debugging kernel - DEBUGGER INSTALLATION CHECK
  AX = 004Fh
Return: AX = F386h if debugger is present
SeeAlso: INT 68/AX=4400h
--------G-410050-----------------------------
INT 41 P - MS Windows debugging kernel - "DefineDebugSegment"
  AX = 0050h
  BX = segment number in executable (0-based)
  CX = selector
  DX = instance handle
```

```
  SI = segment flags (0=code, 1=data)
  ES:DI -> module name of owner
Return: ???
SeeAlso: AX=0012h,AX=004Fh
--------G-410051-----------------------------
INT 41 CPU - MS Windows debugging kernel - MOVE SEGMENT
  AX = 0051h
  ???
Return: ???
SeeAlso: AX=0050h,AX=0052h
--------G-410052-----------------------------
INT 41 CPU - MS Windows debugging kernel - FREE SEGMENT
  AX = 0052h
  BX = freed selector
SeeAlso: AX=0050h,AX=0051h,AX=005Ch
--------G-410059-----------------------------
INT 41 CPU - MS Windows debugging kernel - LOAD TASK
  AX = 0059h
  ???:BX = CS:IP of new task's starting point
--------G-41005C-----------------------------
INT 41 CPU - MS Windows debugging kernel - FREE INITIAL SEGMENT
  AX = 005Ch
  BX = freed selector
Note: called only when KERNEL starts, once for CS and once for the DS alias
    to CS
SeeAlso: AX=0052h
--------G-410060-----------------------------
INT 41 CPU - MS Windows debugging kernel -  END OF SEGMENT LOAD
  AX = 0060h
  ???
Return: ???
SeeAlso: AX=0061h
--------G-410061-----------------------------
INT 41 CPU - MS Windows debugging kernel - END OF SEGMENT DISCARD
  AX = 0061h
  ???
Return: ???
SeeAlso: AX=0060h
--------G-410062-----------------------------
INT 41 CPU - MS Windows debugging kernel - APPLICATION TERMINATING
  AX = 0062h
```

```
STACK:  BYTE  exit code
Return: ???
  STACK unchanged???
SeeAlso: AX=0064h
--------G-410063-----------------------------
INT 41 CPU - MS Windows debugging kernel - ASYNCHRONOUS STOP (Ctrl-Alt-SysReq)
  AX = 0063h
--------G-410064-----------------------------
INT 41 CPU - MS Windows debugging kernel - DLL LOADED
  AX = 0064h
  CX:BX = DLL entry point CS:IP
  SI = module handle
SeeAlso: AX=0062h,AX=0065h
--------G-410065-----------------------------
INT 41 CPU - MS Windows debugging kernel - MODULE REMOVED
  AX = 0065h
  ES = module handle
SeeAlso: AX=0064h
--------G-410066-----------------------------
INT 41 CPU - MS Windows debugging kernel - ERROR
  AX = 0066h
Note: called by LogError()
SeeAlso: AX=0067h
--------G-410067-----------------------------
INT 41 CPU - MS Windows debugging kernel - PARAMETER ERROR
  AX = 0067h
Note: called by LogParamError()
SeeAlso: AX=0066h
--------V-42---------------------------------
INT 42 - VIDEO - RELOCATED DEFAULT INT 10 VIDEO SERVICES (EGA,VGA)
Desc: contains the address of the original INT 10 handler which an EGA+
    video adapter replaces with its own on-board BIOS code
SeeAlso: INT 10/AH=00h,INT 10/AH=0Eh,INT 6D"VGA"
Note: not used by PS/2 built-in VGA or XGA
--------h-42---------------------------------
INT 42 - Z100 - Master 8259 - Timer
SeeAlso: INT 41"Z100",INT 43"Z100"
--------h-42---------------------------------
INT 42 - TI Professional PC - IRQ2
Note: on the TI Pro, IRQ0 is connected to the same pin on the expansion bus
    that IBM connects to IRQ4
```

```
SeeAlso: INT 0C"IRQ4",INT 41"TI Professional",INT 43"TI Professional"
--------b-42--------------------------------
INT 42 - Western Digital WD1002 SuperBIOS - INT 40 CASCADE
Note: if the second WD1002 controller in the system finds INT 40 already in
    use, it uses this vector to cascade to the first controller's BIOS
SeeAlso: INT 40"DISKETTE",INT 47"SuperBIOS"
--------O-42--------------------------------
INT 42 - Acorn BBC Master 512 - "OSBPUT" - WRITE SINGLE BYTE TO FILE
  AL = byte to be written
  BH = file handle
Return: flags destroyed
SeeAlso: INT 40"Acorn",INT 41"Acorn",INT 43"Acorn",INT 47"Acorn",INT 49"Acorn"
--------V-425F33-----------------------------
INT 42 C - Chips & Technologies '65530' BIOS - MODE SET HOOK
  AX = 5F33h
  BL = current width in characters
  BH = curent video mode
  CH = active display page
Return: nothing
Desc: this function is called at the end of a video mode set
Note: the OEM has the option of enabling or disabling this callout, as well
    as specifying whether the callout occurs on INT 15h or INT 42h
SeeAlso: INT 15/AX=5F31h,INT 15/AX=5F35h,INT 10/AX=5F50h,INT 15/AX=5F33h
--------V-427500-----------------------------
INT 42 U - Toshiba laptops - ???
  AX = 7500h
  BL = ??? (00h or 01h)
Return: ???
Note: used by Toshiba utility VCHAD.EXE
SeeAlso: AX=7501h,AX=7503h
--------V-427501-----------------------------
INT 42 U - Toshiba laptop - GET ??? DATA
  AX = 7501h
  DS:DI -> data area to be filled ???
Return: area filled with data ???
Note: used by Toshiba utility VCHAD.EXE
SeeAlso: AX=7500h,AX=7502h,AX=7503h
--------V-427502-----------------------------
INT 42 U - Toshiba laptops - SET ??? DATA
  AX = 7502h
  DS:DI -> data area ???
```

```
Return: ???
Note: used by Toshiba utility VCHAD.EXE
SeeAlso: AX=7501h,AX=7503h
--------V-427503-----------------------------
INT 42 - Toshiba laptops - GET DISPLAY STATUS
  AX = 7503h
Return: AX = 7575h if supported
  CX = 0001h if supported
  BH = display type (00h color, 03h monochrome)
  BL = display state
       01h internal LCD display is active
       02h external VGA display is active
       03h both displays active / DeskStation display mode enabled
    (not possible on all machines)
Note: used by VCHAD.EXE and supported by all Toshiba VGA laptops until about
    1994 (string "TOSHIBA " at F000:E010h should be checked before call)
  no longer supported by T21xx series, use INT 10/AX=5F50h instead
  INT 42 normally points to F000:F065h but may be redirected by QEMM386
SeeAlso: AX=7500h,AX=7504h,INT 10/AX=5F50h,INT 15/AH=C0h
--------V-427504-----------------------------
INT 42 U - Toshiba laptops - ???
  AX = 7504h
  BL = ???
Return: BH = ???
Note: used by Toshiba utility VCHAD.EXE
SeeAlso: AX=7500h,AX=7503h
--------V-43---------------------------------
INT 43 - VIDEO DATA - CHARACTER TABLE (EGA,MCGA,VGA)
Desc: points at graphics data for characters 00h-7Fh of the current font
    in 8x8 dot modes, graphics data for all characters in 8x14 and 8x16
    modes
Note: this is not a callable vector!
SeeAlso: INT 06"no-name",INT 1F"SYSTEM DATA",INT 44"VIDEO"
--------h-43---------------------------------
INT 43 - Z100 - Master 8259 - Slave 8259 input
Note: slave runs in special fully nested mode
SeeAlso: INT 42"Z100",INT 44"Z100"
--------h-43---------------------------------
INT 43 - TI Professional PC - IRQ3 - TIMER1 25ms INTERVAL INTERRUPT
SeeAlso: INT 0B"IRQ3",INT 42"TI Professional",INT 44"TI Professional"
SeeAlso: INT 58"TI Professional"
```

```
--------O-43---------------------------------
INT 43 - Acorn BBC Master 512 - "OSBGET" - READ SINGLE BYTE FROM FILE
  BH = file handle
Return: CF clear if successful
      AL = byte read from file
  CF set on error
SeeAlso: INT 40"Acorn",INT 41"Acorn",INT 42"Acorn",INT 46"Acorn"
--------V-44---------------------------------
INT 44 - VIDEO DATA - ROM BIOS CHARACTER FONT, CHARACTERS 00h-7Fh (PCjr)
Desc: this vector points at graphics data for current character font
SeeAlso: INT 1F"SYSTEM DATA",INT 43"VIDEO"
--------N-44---------------------------------
INT 44 - Novell NetWare - HIGH-LEVEL LANGUAGE API
--------I-44---------------------------------
INT 44 - IBM 3270-PC High Level Language API
  DS:SI -> parameter control block
--------h-44---------------------------------
INT 44 - Z100 - Master 8259 - Serial A
SeeAlso: INT 43"Z100",INT 45"Z100"
--------h-44---------------------------------
INT 44 - TI Professional PC - IRQ4
Note: on the TI Pro, IRQ4 is connected to the same pin on the expansion bus
    that IBM connects to IRQ5
SeeAlso: INT 0D"IRQ5",INT 43"TI Professional",INT 45"TI Professional"
--------v-44---------------------------------
INT 44 - VIRUS - "Lehigh" - ORIGINAL INT 21h VECTOR
SeeAlso: INT 32"VIRUS",INT 60"VIRUS",INT 70"VIRUS",INT 9E"VIRUS"
--------O-4400-------------------------------
INT 44 - Acorn BBC Master 512 - "OSARGS" - GET/SET FILE PARAMS FOR OPEN FILE
  AH = 00h
  AL = function
      00h get current filing system
    Return: AL = filing system (see #03200)
      01h get address of commandline tail
    Return: BX buffer filled with address of command tail in I/O
            processor address space (use INT 4A/AL=05h to
            retrieve)
      FFh flush all files onto secondary storage
  BX -> 4-byte data buffer
Note: the commandline tail is terminated with a carriage return (0Dh)
SeeAlso: INT 40"Acorn",INT 45"Acorn"
```

```
(Table 03200)
Values for BBC Master filing system:
 00h  none
 01h  1200 bps cassette
 02h  300 bps cassette
 03h  ROM FS
 04h  DFS
 05h  ANFS/NFS
 06h  TFS
 08h  ADFS
--------O-44----------------------------------
INT 44 - Acorn BBC Master 512 - "OSARGS" - GET/SET FILE PARAMS FOR OPEN FILE
  AH = nonzero file handle
  AL = function
       00h get sequential pointer for file
       01h set sequential pointer for file
       02h get length of file
  BX -> 4-byte data buffer
Return: BX buffer updated if appropriate
SeeAlso: INT 40"Acorn",INT 41"Acorn",INT 44/AH=00h,INT 45"Acorn",INT 4A"Acorn"
--------h-45----------------------------------
INT 45 - Z100 - Master 8259 - Serial B
SeeAlso: INT 44"Z100",INT 46"Z100"
--------h-45----------------------------------
INT 45 - TI Professional PC - IRQ5
Note: on the TI Pro, IRQ5 is connected to the same pin on the expansion bus
    that IBM connects to IRQ6
SeeAlso: INT 0E"IRQ6",INT 44"TI Professional",INT 46"TI Professional"
--------O-45----------------------------------
INT 45 - Acorn BBC Master 512 - "OSFILE" - READ/WRITE FILE OR DIRECTORY INFO
  AL = function
       00h save block of memory as file
       01h update directory entry for existing file
       02h set load address for existing file
       03h set execution address for existing file
       04h set attributes for existing file
       05h read directory
       06h delete file
       FFh load file
  DS:BX -> control block (see #03201)
```

```
Return: FLAGS destroyed
  AL = file type
       00h not found
       01h file found
       02h directory found
       FFh protected file
SeeAlso: INT 40"Acorn",INT 41"Acorn",INT 44"Acorn",INT 46"Acorn"

Format of BBC Master control block:
Offset  Size  Description (Table 03201)
 00h  WORD  address of CR-terminated filename
 02h  DWORD load address of file
 06h  DWORD execution address of file
 0Ah  DWORD start address of data to save
 0Eh  DWORD end address of data to save, or file attributes
    file attributes in low byte (see #03202)
    other three bytes are filing-system specific file attributes

Bitfields for BBC Master file attributes:
Bit(s)  Description (Table 03202)
 0  no owner read access
 1  no owner write access
 2  not executable by owner
 3  not deletable by owner
 4  no public read access
 5  no public write access
 6  not executable with public access
 7  not deletable with public access
--------B-46---------------------------------
INT 46 - SYSTEM DATA - HARD DISK 1 DRIVE PARAMETER TABLE ADDRESS [NOT A VECTOR!]
Note: not used by some PS/2 models
SeeAlso: INT 13/AH=09h,INT 41"HARD DISK 0",INT 60"Adaptec",INT C0"AMI"
--------h-46---------------------------------
INT 46 - Z100 - Master 8259 - Keyboard, Retrace, and Light Pen
SeeAlso: INT 45"Z100",INT 47"Z100"
--------h-46---------------------------------
INT 46 - TI Professional PC - IRQ6 - FLOPPY DISK CONTROLLER
Note: on the TI Pro, IRQ6 is connected to the same pin on the expansion bus
    that IBM connects to IRQ7
SeeAlso: INT 0F"IRQ7",INT 45"TI Professional",INT 47"TI Professional"
--------O-46---------------------------------
```

```
INT 46 - Acorn BBC Master 512 - "OSRDCH" - GET CHARACTER FROM CUR INPUT STREAM
Return: CF clear if successful
      AL = character read
  CF set on error
      AL = error code
SeeAlso: INT 40"Acorn",INT 43"Acorn",INT 47"Acorn",INT 49"Acorn"
--------h-47---------------------------------
INT 47 - Z100 - Master 8259 - Printer
SeeAlso: INT 46"Z100",INT 48"Z100"
--------h-47---------------------------------
INT 47 - TI Professional PC - IRQ7 - KEYBOARD USART
SeeAlso: INT 09"IRQ1",INT 46"TI Professional"
--------O-47---------------------------------
INT 47 - Acorn BBC Master 512 - "OSWRCH" - WRITE CHARACTER TO CUR OUTPUT STREAM
  AL = character to be written
Return: FLAGS destroyed
SeeAlso: INT 40"Acorn",INT 46"Acorn",INT 49"Acorn"
--------b-47---------------------------------
INT 47 - Western Digital WD1002-27X SuperBIOS - INT 40 CASCADE
Desc: used by the second WD1002-27X controller to cascade to the first
    controller's INT 40
SeeAlso: INT 40"DISKETTE",INT 42"SuperBIOS",INT 48"SuperBIOS"
----------478000-----------------------------
INT 47 - SQL Base - DATABASE ENGINE API
  AX = 8000h
  DS:BX -> parameter block, first word is function number (see #03203)
Program: SQL Base is a network-oriented database engine by Gupta Technologies
SeeAlso: AX=8001h

(Table 03203)
Values for SQL Base function number:
 01h  "SQLFINI" initalialize application's use of the database
 02h  "SQLFDON" application is done using the database
 03h  "SQLFCON" connect to a cursor/database
 04h  "SQLFDIS" disconnect from a cursor/database
 05h  "SQLFCOM" compile a SQL command
 06h  "SQLFEXE" execute a SQL command
 07h  "SQLFCEX" compile and execute a SQL command
 08h  "SQLFCMT" commit a transaction to the database
 09h  "SQLFDES" describe the items of a SELECT statement
 0Ah  "SQLFGFI" get fetch information
```

```
0Bh  "SQLFFBK" fetch previous result row from SELECT statement
0Ch  "SQLFFET" fetch next result row from SELECT statement
0Dh  "SQLFEFB" enable fetch backwards
0Eh  "SQLFPRS" position in result set
0Fh  "SQLFURS" undo result set
10h  "SQLFNBV" get number of bind variables
11h  "SQLFBND" bind data variables
12h  "SQLFBNN" bind numerics
13h  "SQLFBLN" bind long number
14h  "SQLFBLD" bind long data variables
15h  "SQLFSRS" start restriction set processing
16h  "SQLFRRS" restart restriction set processing
17h  "SQLFCRS" close restriction set
18h  "SQLFDRS" drop restriction set
19h  "SQLFARF" apply Roll Forward journal
1Ah  "SQLFERF" end Roll Forward journal
1Bh  "SQLFSRF" start Roll Forward journal
1Ch  "SQLFSTO" store a compiled SQL command
1Dh  "SQLFRET" retrieve a compiled SQL command
1Eh  "SQLFDST" drop a stored command
1Fh  "SQLFCTY" get command type
20h  "SQLFEPO" get error position
21h  "SQLFGNR" get number of rows
22h  "SQLFNSI" get number of select items
23h  "SQLFRBF" get Roll Back flag
24h  "SQLFRCD" get return code
25h  "SQLFROW" get number of ROWs
26h  "SQLFSCN" set cursor name
27h  "SQLFSIL" set isolation level
28h  "SQLFSLP" set log parameters
29h  "SQLFSSB" set select buffer
2Ah  "SQLFSSS" set sort space
2Bh  "SQLFRLO" read long
2Ch  "SQLFWLO" write long
2Dh  "SQLFLSK" long seek
2Eh  "SQLFGLS" get long size
2Fh  "SQLFELO" end long operation
30h  "SQLFRBK" roll back a transaction from the database
31h  "SQLFERR" error message
32h  "SQLFCPY" copy
33h  "SQLFR01" reserved
```

```
 34h  "SQLFSYS" system
 35h  "SQLFSTA" statistics
 36h  "SQLFR02" reserved
 37h  "SQLFXAD" extra add
 38h  "SQLFXCN" extra character to number
 39h  "SQLFXDA" extra date add
 3Ah  "SQLFXDP" extra date picture
 3Bh  "SQLFXDV" extra divide
 3Ch  "SQLFXML" extra multiply
 3Dh  "SQLFXNP" extra number picture
 3Eh  "SQLFXPD" extra picture date
 3Fh  "SQLFXSB" extra subtract
 40h  "SQLFINS" install database
 41h  "SQLFDIN" deinstall database
 42h  "SQLFDIR" directory of databases
 43h  "SQLFTIO" timeout
 44h  "SQLFFQN" get fully qualified column name
 45h  "SQLFEXP" explain execution plan
 46h  "SQLFFER" get full error
 47h  "SQLFBKP" begin online backup
 48h  "SQLFRDC" read backup data chunk
 49h  "SQLFEBK" end backup
 4Ah  "SQLFRES" begin restore from backup
 4Bh  "SQLFWDC" write backup data chunk for restore
 4Ch  "SQLFRRD" recover restored database to consistent state
 4Dh  "SQLFERS" end restore
 4Eh  "SQLFNRR" return number of result set rows
 4Fh  "SQLFSTR" start restriction mode
 50h  "SQLFSPR" stop restriction mode
 51h  "SQLFCNC" connect 2
 52h  "SQLFCNR" connect with no recovery
 53h  "SQLFOMS" set output message size
 54h  "SQLFIMS" set input message size
 55h  "SQLFSCP" set cache pages
 56h  "SQLFDSC" describe items of a SELECT statement (external)
 57h  "SQLFLAB" get label info for items in SELECT statement
 58h  "SQLFCBV" clear bind variables
 59h  "SQLFGET" get database information
 5Ah  "SQLFSET" set database information
 5Bh  "SQLFTEC" translate error code
----------478001-----------------------------
```

```
INT 47 - SQL Base - GET VERSION NUMBER
  AX = 8001h
Return: ???
Program: SQL Base is a network-oriented database engine by Gupta Technologies
SeeAlso: AX=8000h
--------B-48---------------------------------
INT 48 - KEYBOARD - CORDLESS KEYBOARD TRANSLATION (PCjr)
  AL = scan code???
Note: This interrupt may be un-initialized (0000h:0000h) on old machines.
    This should be checked before calling or hooking this vector.
  MS-DOS/PC DOS 3.3x-4.x KEYB hooked the INT 48h handler. For AL <= 80h
    it checked that either ALT and neither CTRL key was pressed,
    and in that case, it cleared the CTRL flag in the BIOS variable at
    0040:0017h and stored its contents in an internal variable,
    before continuing with the previous (non-zero) INT 48h handler.
SeeAlso: INT 49"PCjr"
--------h-48---------------------------------
INT 48 - Z100 - Slave 8259 - S100 vectored line 0
SeeAlso: INT 47"Z100",INT 49"Z100"
--------N-48---------------------------------
INT 48 - Watstar PC Network data pointer 1
SeeAlso: INT 49"Watstar"
--------O-48---------------------------------
INT 48 - Acorn BBC Master 512 - "OSNEWL" - SEND NEWLINE TO OUTPUT STREAM
Return: FLAGS destroyed
Note: writes a carriage return (0Dh) followed by a linefeed (0Ah)
SeeAlso: INT 40"Acorn",INT 47"Acorn",INT 49"Acorn"
--------b-48---------------------------------
INT 48 - Western Digital WD1002-27X SuperBIOS - DRIVE DATA (NOT A VECTOR!)
Note: the second WD1002-27X controller in a system uses the low byte to
    store the number of drives controlled by the second controller,
    and the high word for temporary storage during track recalculation;
    the first controller uses offsets 74h-77h in the BIOS data area
    (refer to MEMORY.LST) to store data
SeeAlso: INT 47"SuperBIOS"
--------V-48---------------------------------
INT 48 U - Compaq UILIB.EXE - API
  AX = function (see #03204)
  BX = call type (0002h) (see #03207)
  ???
Return: ???
```

```
Note: returns AX=FFFFh if 1000h<=AX<=2000h and AX is not one of the functions
    listed below
SeeAlso: AX=1A70h

(Table 03204)
Values for valid UILIB function number:
 1000h  1160h 12D0h 1430h 1570h 1680h 17F0h 1920h 1A90h
 1010h  1170h 12E0h 1440h 1578h 1690h 1800h 1930h 1AA0h
 1020h  1180h 12F0h 1450h 1580h 16A0h 1810h 1940h
 1030h  1190h 1300h 1460h 1590h 16B0h 1820h 1950h
 1040h  11A0h 1310h 1470h 1594h 16C0h 1830h 1960h
 1050h  11B0h 1320h 1480h 1598h 16D0h 1840h 1970h
 1060h  11C0h 1330h 1490h 15A0h 16E0h 1848h 1980h
 1070h  11D0h 1340h 14A0h 15B0h 16F0h 1850h 1990h
 1080h  11E0h 1350h 14B0h 15C0h 1700h 1860h 19A0h
 1090h  11F0h 1360h 14B8h 15D0h 1710h 1870h 19B0h
 1095h  1200h 1370h 14BBh 15D4h 1720h 1878h 19C0h
 1098h  1210h 1380h 14C0h 15D8h 1730h 1880h 19D0h
 10A0h  1220h 1390h 14D0h 15E0h 1735h 1890h 19E0h
 10C0h  1230h 13A0h 14E0h 15F0h 1740h 1898h 19F0h
 10D0h  1240h 13B0h 14F0h 1600h 1750h 18A0h 1A00h
 10E0h  1250h 13B8h 1500h 1610h 1770h 18B0h 1A10h
 10F0h  1260h 13C0h 1508h 1620h 1780h 18C0h 1A20h
 1100h  1270h 13D0h 1510h 1630h 1790h 18D0h 1A30h
 1110h  1280h 13E0h 1520h 1640h 17A0h 18E0h 1A40h
 1120h  1290h 13F0h 1530h 1650h 17B0h 18F0h 1A50h
 1130h  12A0h 1400h 1540h 1660h 17C0h 1900h 1A60h
 1140h  12B0h 1410h 1550h 1664h 17D0h 1909h 1A70h
 1150h  12C0h 1420h 1560h 1670h 17E0h 1910h 1A80h
--------b-4800--------------------------------
INT 48 - TI Professional PC - SPEAKER DEVICE - SOUND SPEAKER
  AH = 00h
  AL = number of 25ms ticks sound should last
Return: nothing
Desc: sound the speaker at the current frequency setting (see AH=02h) for
    the indicated duration
Notes:  this function returns immediately; the sound is terminated by the
    timer interrupt handler
  if a new sound is requested while one is already in progress, the
    previous sound is terminated immediately and the new sound takes
    its place
```

```
SeeAlso: AH=01h,AH=02h,AH=03h,AH=04h,AH=06h,AH=08h,AH=0Ah,AH=0Bh
SeeAlso: INT 40"TI Professional",INT 49/AH=01h"TI"
SeeAlso: INT 4A/AH=00h"TI",INT 4C"TI Professional",INT 4D/AH=00h
--------b-4801--------------------------------
INT 48 - TI Professional PC - SPEAKER DEVICE - CHECK SPEAKER STATUS
  AH = 01h
Return: ZF clear if speaker is currently on
  ZF set if speaker is currently off
SeeAlso: AH=00h,AH=02h,AH=03h,AH=04h,AH=06h,AH=08h,AH=0Ah,AH=0Bh
--------b-4802--------------------------------
INT 48 - TI Professional PC - SPEAKER DEVICE - SET SPEAKER FREQUENCY
  AH = 02h
  CX = frequency divisor (freq = 1250000 / CX)
Return: nothing
SeeAlso: AH=00h,AH=01h,AH=03h,AH=04h,AH=06h,AH=08h,AH=0Ah,AH=0Bh
--------b-4803--------------------------------
INT 48 - TI Professional PC - SPEAKER DEVICE - TURN ON SPEAKER
  AH = 03h
Return: nothing
Desc: turn on the speaker at the current frequency, leaving it on until
    explicitly turned off with AH=04h or the end of a subsequent
    AH=00h
SeeAlso: AH=00h,AH=01h,AH=02h,AH=04h,AH=06h,AH=08h,AH=0Ah,AH=0Bh
--------b-4804--------------------------------
INT 48 - TI Professional PC - SPEAKER DEVICE - TURN OFF SPEAKER
  AH = 04h
Return: nothing
SeeAlso: AH=00h,AH=01h,AH=02h,AH=03h,AH=04h,AH=06h,AH=08h,AH=0Ah,AH=0Bh
--------b-4805--------------------------------
INT 48 - TI Professional PC - SPEAKER DEVICE - DELAY
  AH = 05h
  CX = desired delay in milliseconds
Return: after delay expires
Note: the delay is only approximate, and may be longer than requested
SeeAlso: AH=00h,AH=01h,AH=02h,AH=03h,AH=04h,AH=06h,AH=08h,AH=0Ah,AH=0Bh
--------b-4806--------------------------------
INT 48 - TI Professional PC - CALCULATE CRC
  AH = 06h
  ES:BX -> memory block for which to calculate CRC
  BP = size of block in bytes
Return: DX = CRC for block
```

```
  ZF set if DX = 0000h
Note: if the CRC of a memory block is appended to the block, then the CRC
    of the block plus CRC should equal 0000h
SeeAlso: AH=00h,AH=01h,AH=02h,AH=03h,AH=04h,AH=06h,AH=08h,AH=0Ah,AH=0Bh
--------b-4807--------------------------------
INT 48 - TI Professional PC - PRINT ROM MESSAGE
  AH = 07h
  SI = offset of ASCIZ message string within segment F400h
Return: nothing
SeeAlso: AH=00h,AH=01h,AH=02h,AH=03h,AH=04h,AH=06h,AH=08h,AH=0Ah,AH=0Bh
--------b-4808--------------------------------
INT 48 - TI Professional PC - DISPLAY SYSTEM ERROR MESSAGE
  AH = 08h
  BX = error number
Return: nothing
Desc: displays the error message " ** System Error ** - xxxx" where xxxx is
    the hexadecimal value in BX
SeeAlso: AH=00h,AH=01h,AH=02h,AH=03h,AH=04h,AH=06h,AH=08h,AH=0Ah,AH=0Bh
--------b-4809--------------------------------
INT 48 - TI Professional PC - GET SYSTEM CONFIGURATION DATA
  AH = 09h
Return: ES:BX -> system configuration word (see #03227)
SeeAlso: AH=00h,AH=01h,AH=02h,AH=03h,AH=04h,AH=06h,AH=08h,AH=0Ah,AH=0Bh
--------b-480A--------------------------------
INT 48 - TI Professional PC - GET EXTRA SYSTEM CONFIGURATION INFO ADDRESS
  AH = 0Ah
Return: ES:BX -> configuration information (see #03205)
SeeAlso: AH=00h,AH=01h,AH=02h,AH=03h,AH=04h,AH=06h,AH=08h,AH=0Bh

Format of TI Professional PC extra system configuration information:
Offset  Size  Description (Table 03205)
 -3 WORD  memory size in paragraphs
 00h  BYTE  drive type byte (see #03206)
 01h  WORD  extra system configuration word 1
    bit 0: 8087 is present
    bits 15-1: reserved (0)
 03h  WORD  extra system configuration word 2
    bits 15-0: reserved (0)

Bitfields for TI Professional PC drive type byte:
Bit(s)  Description (Table 03206)
```

```
 0  drive A is double-sided
 1  drive A has 80 tracks instead of 40
 2  drive B is double-sided
 3  drive B has 80 tracks instead of 40
 4  drive C is double-sided
 5  drive C has 80 tracks instead of 40
 6  drive D is double-sided
 7  drive D has 80 tracks instead of 40
Note: the type for drive A is determined by motherboard switches; the
    remaining drives' types are set from a table in IO.SYS
SeeAlso: #03205
--------b-480B--------------------------------
INT 48 - TI Professional PC - GET EXTRA SYSTEM CONFIGURATION INFORMATION
  AH = 0Bh
Return: AL = drive type byte (see #03206)
  BX = extra system configuration word 1 (see #03205)
  CX = extra system configuration word 2 (see #03205)
  AH destroyed
SeeAlso: AH=00h,AH=01h,AH=02h,AH=03h,AH=04h,AH=06h,AH=08h,AH=0Ah
--------V-481A70-----------------------------
INT 48 U - Compaq UILIB.EXE - INSTALLATION CHECK
  AX = 1A70h
  BX = call type (see #03207)
Return: CX = 5649h ('VI') if installed
  DX = 4557h ('EW') if installed
      AX = version??? (0106h)

(Table 03207)
Values for UILIB call type:
 0000h  near
 0001h  far
 0002h  INT (only valid call type when using INT 48)
 0003h  near
--------B-49---------------------------------
INT 49 - SYSTEM DATA - NON-KEYBOARD SCAN-CODE TRANSLATION TABLE (PCjr)
SeeAlso: #03208,INT 48"PCjr"

Format of PCjr scan-code translation table:
Offset  Size  Description (Table 03208)
 00h  BYTE  number of non-keyboard scancodes in the table
 01h  N WORDs high byte 00h (NUL) byte scancode with low order byte
```

```
      representing the scancode mapped values relative to their
      input values within the range of 56h through 7Eh
--------h-49--------------------------------
INT 49 - Z100 - Slave 8259 - S100 vectored line 1
SeeAlso: INT 48"Z100",INT 4A"Z100"
--------N-49--------------------------------
INT 49 - Watstar PC Network data pointer 2
SeeAlso: INT 48"Watstar"
--------O-49--------------------------------
INT 49 - Acorn BBC Master 512 - "OSASCI" - WRITE CHARACTER TO CUR OUTPUT STREAM
  AL = character to be written
Return: FLAGS destroyed
Note: converts carriage return (0Dh) into CRLF sequence (0Dh 0Ah)
SeeAlso: INT 40"Acorn",INT 46"Acorn",INT 47"Acorn",INT 48"Acorn"
--------b-49--------------------------------
INT 49 - Tandy 2000 - BOOTSTRAP LOADER
Note: this interrupt is identical to INT 19
SeeAlso: INT 19,INT 4A"Tandy 2000",INT 4C"Tandy 2000",INT 51"Tandy 2000"
--------a-490001----------------------------
INT 49 - MAGic v1.16+ - TURN ON MAGNIFICATION
  AX = 0001h
Return: AX = status (see #03209)
  BX,CX,DX destroyed
Program: MAGic (MAGnification In Color) is a TSR by Microsystems Software, Inc.
    providing 2x2 text and graphics magnification on VGA, XGA, and SVGA
Note: INT 49 is the default, but may be overridden on the commandline.  The
    actual interrupt in use may be found by searching for the signature
    "MAGic" or "xMAGic" (for the deluxe version) immediately preceding
    the interrupt handler (this is also the installation check).  MAGic
    uses CodeRunneR, which places the signature "RT" at offset 0000h in
    the interrupt handler's segment, followed by MAGic's TSR ID of
    "VMAG".
SeeAlso: AX=0002h,AX=0003h,AX=0004h,AX=0008h
Index:  installation check;MAGic

(Table 03209)
Values for MAGic status:
 0000h  cannot magnify current video mode
 0002h  magnified (text mode)
 0003h  magnified (graphics mode)
 FFFDh  function works only in magnified mode
```

```
 FFFFh  MAGic busy, retry later
--------a-490002-----------------------------
INT 49 - MAGic v1.16+ - TURN OFF MAGNIFICATION
  AX = 0002h
Return: AX = status (see #03209)
  BX,CX,DX destroyed
SeeAlso: AX=0001h
--------a-490003-----------------------------
INT 49 - MAGic v1.16+ - SHIFT MAGNIFIED WINDOW TO INCLUDE SPECIFIED LOCATION
  AX = 0003h
  BX = vertical position (character row [text] or pixel row [graphics])
  DX = horizontal position (char column [text] or 8-pixel units [gr])
Return: AX = status
      0000h successful
      FFFFh MAGic busy, retry later
  BX,CX,DX destroyed
Note: window is not moved if the position is inside the current window
SeeAlso: AX=0001h,AX=0004h,AX=0005h
--------a-490004-----------------------------
INT 49 - MAGic v1.16+ - REPOSITION MAGNIFIED WINDOW
  AX = 0004h
  BX = vertical position of upper left corner
  DX = horizontal position
Return: AX = status (see AX=0003h)
  BX,CX,DX destroyed
SeeAlso: AX=0001h,AX=0003h,AX=0005h
--------a-490005-----------------------------
INT 49 - MAGic v1.16+ - GET POSITION OF MAGNIFIED WINDOW
  AX = 0005h
Return: AX = status
      0000h successful
    BX = vertical position (char row or pixel row)
    DX = horizontal position (char column or 8-pixel units)
      FFFFh MAGic busy, retry later
    BX,DX destroyed
  CX destroyed
SeeAlso: AX=0001h,AX=0003h,AX=0004h,AX=0006h,AX=0007h
--------a-490006-----------------------------
INT 49 - MAGic v1.16+ - GET SIZE OF FULL SCREEN
  AX = 0006h
Return: AX = status
```

```
      0000h successful
    BX = vertical size (char rows or pixel rows)
    DX = horizontal size (char cols or 8-pixel units)
      FFFFh MAGic busy, retry later
    BX,DX destroyed
  CX destroyed
SeeAlso: AX=0001h,AX=0005h,AX=0007h
--------a-490007-----------------------------
INT 49 - MAGic v1.16+ - GET SIZE OF MAGNIFICATION WINDOW
  AX = 0007h
Return: AX = status
      0000h successful
    BX = vertical size (char rows or pixel rows)
    DX = horizontal size (char cols or 8-pixel units)
      FFFEh invalid function
      FFFFh MAGic busy, retry later
    BX,DX destroyed
  CX destroyed
BUG:  in v1.16 and v1.17, this function is not recognized as valid, but
    AX=0000h is accepted and will branch into hyperspace
SeeAlso: AX=0001h,AX=0006h
--------a-490008-----------------------------
INT 49 - MAGic v1.23+ - SET TEXT MODE MAGNIFICATION SIZE
  AX = 0008h
  BX = scaling factor (01h=1.4 times, 02h, 04h, 06h, 08h, 09h=12 times)
Return: AX = status
      0000h successful
      FFFBh scaling factor only available in MAGic Deluxe
      FFFCh already in magnified state, can't set size
Notes:  this call specifies the amount a subsequent call to AX=0001h should
    magnify the display
  scaling factors greater than 2 are only available in MAGic Deluxe
SeeAlso: AX=0001h
--------V-4901-------------------------------
INT 49 - TI Professional PC - CRT - SET CURSOR SIZE AND TYPE
  AH = 01h
  CH = cursor start line (bits 3-0) and status (bits 6-5)
      status bits:
    00 non-blinking cursor
    01 no cursor
    10 fast-blinking cursor
```

```
    11 slow-blinking cursor
  CL = cursor end line
Return: nothing
Note: AH=00h,04h,05h,0Bh,0Ch,0Dh,0Fh are documented as NOPs
SeeAlso: AH=02h,AH=03h,INT 40"TI Professional",INT 48/AH=00h"TI Professional"
SeeAlso: INT 4A/AH=00h"TI",INT 4B"TI Professional",INT 4D/AH=00h
SeeAlso: INT 57"TI Professional"
--------V-4902-------------------------------
INT 49 - TI Professional PC - CRT - SET CURSOR POSITION
  AH = 02h
  DH = column
  DL = row
Return: DX destroyed
Notes:  AH=00h,04h,05h,0Bh,0Ch,0Dh,0Fh are documented as NOPs
  the TI swaps the row and column compared to the equivalent IBM call
SeeAlso: AH=01h,AH=03h
--------V-4903-------------------------------
INT 49 - TI Professional PC - CRT - GET CURSOR POSTION AND TYPE
  AH = 03h
Return: CH = cursor start and status (see AH=01h)
  CL = cursor end line
  DH = cursor column
  DL = cursor row
Note: AH=00h,04h,05h,0Bh,0Ch,0Dh,0Fh are documented as NOPs
SeeAlso: AH=01h,AH=02h
--------V-4906-------------------------------
INT 49 - TI Professional PC - CRT - SCROLL UP/COPY WINDOW
  AH = 06h
  AL = source blanking
       00h blank source region (move/scroll)
       nonzero do not blank source region (copy)
  DH,DL = source start column,row
  BH,BL = destination start column,row
  CH = width of region to move/copy
  CL = height of region to move/copy
Return: nothing
Notes:  AH=00h,04h,05h,0Bh,0Ch,0Dh,0Fh are documented as NOPs
  the specified region may be wider than the screen, but reliable
    operation then requires that the height be exactly one row
SeeAlso: AH=01h,AH=02h,AH=07h,AH=13h,AH=14h
--------V-4907-------------------------------
```

```
INT 49 - TI Professional PC - CRT - SCROLL DOWN/COPY WINDOW
  AH = 07h
  AL = source blanking
       00h blank source region (move/scroll)
       nonzero do not blank source region (copy)
  DH,DL = source start column,row
  BH,BL = destination start column,row
  CH = width of region to move/copy
  CL = height of region to move/copy
Return: nothing
Note: AH=00h,04h,05h,0Bh,0Ch,0Dh,0Fh are documented as NOPs
SeeAlso: AH=01h,AH=02h,AH=06h,AH=13h,AH=14h
--------V-4908--------------------------------
INT 49 - TI Professional PC - CRT - GET CHARACTER AND ATTRIBUTE AT POSITION
  AH = 08h
Return: AL = character at current cursor position
  AH = attribute
Note: AH=00h,04h,05h,0Bh,0Ch,0Dh,0Fh are documented as NOPs
SeeAlso: AH=01h,AH=09h,AH=0Ah,AH=0Eh,INT 10/AH=08h
--------V-4909--------------------------------
INT 49 - TI Professional PC - CRT - WRITE CHARACTER(S) WITH ATTRIBUTE
  AH = 09h
  AL = character to write
  BL = attribute to use (becomes new current attribute)
  CX = number of times to write character
Return: nothing
Desc: write CX copies of the character in AL beginning at the current cursor
    position
Note: AH=00h,04h,05h,0Bh,0Ch,0Dh,0Fh are documented as NOPs
SeeAlso: AH=01h,AH=08h,AH=0Ah,AH=0Eh,INT 10/AH=09h
--------V-490A--------------------------------
INT 49 - TI Professional PC - CRT - WRITE CHARACTER(S) WITH CURRENT ATTRIBUTE
  AH = 0Ah
  AL = character to write
  CX = number of times to write character
Return: nothing
Desc: write CX copies of the character in AL beginning at the current cursor
    position
Note: AH=00h,04h,05h,0Bh,0Ch,0Dh,0Fh are documented as NOPs
SeeAlso: AH=01h,AH=02h,AH=08h,AH=09h,AH=0Eh,INT 10/AH=0Ah
--------V-490E--------------------------------
```

```
INT 49 - TI Professional PC - CRT - TTY OUTPUT
  AH = 0Eh
  AL = character to write
Return: nothing
Desc: write the character in AL at the current cursor position, advancing
    the cursor, and interpreting CR, LF, TAB, and BEL characters
Note: AH=00h,04h,05h,0Bh,0Ch,0Dh,0Fh are documented as NOPs
SeeAlso: AH=01h,AH=02h,AH=08h,AH=09h,AH=0Ah,INT 10/AH=0Eh
--------V-4910--------------------------------
INT 49 - TI Professional PC - CRT - WRITE BLOCK OF CHARACTERS WITH ATTRIBUTE
  AH = 10h
  AL = attribute (becomes new current attribute)
  DX:BX -> string of characters to write
  CX = length of string
Return: nothing
Note: AH=00h,04h,05h,0Bh,0Ch,0Dh,0Fh are documented as NOPs
BUG:  CX must not be 0000h on entry, or the system will crash
SeeAlso: AH=01h,AH=02h,AH=09h,AH=0Eh,AH=11h
--------V-4911--------------------------------
INT 49 - TI Professional PC - CRT - WRITE BLOCK OF CHARACTERS WITH CURR ATTRIB
  AH = 11h
  DX:BX -> string of characters to write
  CX = length of string
Return: nothing
Note: AH=00h,04h,05h,0Bh,0Ch,0Dh,0Fh are documented as NOPs
BUG:  CX must not be 0000h on entry, or the system will crash
SeeAlso: AH=01h,AH=02h,AH=09h,AH=0Eh,AH=10h
--------V-4912--------------------------------
INT 49 - TI Professional PC - CRT - FILL ENTIRE SCREEN WITH ATTRIBUTE
  AH = 12h
  AL = attribute (see #03210)
Return: nothing
Note: AH=00h,04h,05h,0Bh,0Ch,0Dh,0Fh are documented as NOPs
SeeAlso: AH=01h,AH=02h

Bitfields for TI Professional PC screen attribute:
Bit(s)  Description (Table 03210)
 7  alternate character set (requires user-supplied ROM)
 6  blink
 5  underline
 4  reverse video
```

```
 3  character enable
 2  green (color) or 58% intensity (gray-scale)
 1  red (color)  or 27.5% intensity
 0  blue (color) or 14.5% intensity
--------V-4913--------------------------------
INT 49 - TI Professional PC - CRT - CLEAR ENTIRE TEXT SCREEN AND HOME CURSOR
  AH = 13h
Return: nothing
Note: AH=00h,04h,05h,0Bh,0Ch,0Dh,0Fh are documented as NOPs
SeeAlso: AH=01h,AH=02h,AH=06h,AH=14h
--------V-4914--------------------------------
INT 49 - TI Professional PC - CRT - CLEAR ENTIRE GRAPHICS SCREEN
  AH = 14h
Return: nothing
Note: AH=00h,04h,05h,0Bh,0Ch,0Dh,0Fh are documented as NOPs
SeeAlso: AH=01h,AH=02h,AH=06h,AH=13h
--------V-4915--------------------------------
INT 49 - TI Professional PC - CRT - SET PROTECTED STATUS AREA
  AH = 15h
  CL = row at which to start status area, or 00h to cancel
  CH = 00h
Return: nothing
Desc: set a protected area of the screen which will not be affected by TTY
    writes or the scrolls they may generate
Notes:  AH=00h,04h,05h,0Bh,0Ch,0Dh,0Fh are documented as NOPs
  the current cursor position must be above the status area in order to
    set the protected area
SeeAlso: AH=01h,AH=02h
--------V-4916--------------------------------
INT 49 - TI Professional PC - CRT - SET ATTRIBUTE LATCH
  AH = 16h
  BL = new attribute (see #03210)
Return: nothing
Note: AH=00h,04h,05h,0Bh,0Ch,0Dh,0Fh are documented as NOPs
SeeAlso: AH=01h,AH=02h
--------V-4917--------------------------------
INT 49 - TI Professional PC - CRT - GET START-OF-DISPLAY POINTER
  AH = 17h
Return: DX = current offset at which display starts
Note: AH=00h,04h,05h,0Bh,0Ch,0Dh,0Fh are documented as NOPs
SeeAlso: AH=01h,AH=02h,INT 10/AH=FEh
```

```
--------V-4918--------------------------------
INT 49 - TI Professional PC - CRT - PRINT TTY STRING
  AH = 18h
  CS:BX -> counted string (count byte with length followed by string)
Return: nothing
Notes:  AH=00h,04h,05h,0Bh,0Ch,0Dh,0Fh are documented as NOPs
  the string must be located in the caller's code segment; any TSRs
    which want to hook INT 49 must check for this function and emulate
    it, because the BIOS retrieves the caller's CS from the stack
SeeAlso: AH=01h,AH=02h,AH=0Eh
--------B-4A---------------------------------
INT 4A C - SYSTEM - USER ALARM HANDLER
Desc: This interrupt is invoked by the BIOS when a real-time clock alarm
    occurs; an application may use it to perform an action at a
    predetermined time.
Note: this interrupt is called from within a hardware interrupt handler,
    so all usual precautions against reentering DOS must be taken
SeeAlso: INT 1A/AH=06h
--------h-4A---------------------------------
INT 4A - Z100 - Slave 8259 - S100 vectored line 2
SeeAlso: INT 49"Z100",INT 4B"Z100"
--------b-4A---------------------------------
INT 4A - Tandy 2000 - PRINT SCREEN
Note: this interrupt is identical to INT 05
SeeAlso: INT 05"PRINT SCREEN"
--------O-4A---------------------------------
INT 4A - Acorn BBC Master 512 - "OSWORD" - MISC FUNCTIONS USING CONTROL BLOCK
  AL = function code
      FAh transfer data between 80186 and 65C12 I/O processor
  DS:BX -> control block (see #03211)
Return: FLAGS destroyed
  control block updated
Note: there are more functions than are listed here, but details are not
    available
SeeAlso: INT 40"Acorn",INT 4B"Acorn",INT 4C"Acorn"

Format of BBC Master control block for function FAh:
Offset  Size  Description (Table 03211)
 00h  BYTE  number of parameters sent to I/O processor (0Dh,0Eh)
 01h  BYTE  number of parameters read from I/O processor (01h)
 02h  DWORD I/O processor address
```

```
 06h  DWORD 80186 segment:offset address
 0Ah  WORD  number of bytes to transfer
 0Ch  BYTE  operation type
    00h write to 65C12 at 24 us/byte
    01h read from 65C12 at 24 us/byte
    02h write to 65C12 at 26 us/word
    03h read from 65C12 at 26 us/word
    04h write to 65C12 at 10 us/byte using 256-byte blocks
    05h read from 65C12 at 10 us/byte using 256-byte blocks
 0Dh  BYTE  65C12 memory access control (only used if offset 00h = 0Eh)
    (see #03212)

Bitfields for 65C12 memory access control:
Bit(s)  Description (Table 03212)
 7  unused
 6  always use main screen memory if I/O addr 3000h-7FFFh (overrides bit 5)
 5  use shadow screen memory if screen address specified
 4  use current ROM rather than ROM selected by bits 3-0 (only if I/O
    address between 8000h and BFFFh)
 3-0  paged ROM number
--------b-4A00--------------------------------
INT 4A - TI Professional PC - KEYBOARD - GET KEYPRESS
  AH = 00h
Return: AX = keystroke (AH=00h for ASCII keys -- no scan code)
SeeAlso: AH=01h,AH=02h,AH=03h,AH=04h,AH=05h,INT 16/AH=00h
SeeAlso: INT 47"TI Professional",INT 48/AH=00h"TI Professional"
SeeAlso: INT 49/AH=01h"TI",INT 4C"TI Professional",INT 4D/AH=00h
SeeAlso: INT 5B"TI Professional"
--------b-4A01--------------------------------
INT 4A - TI Professional PC - KEYBOARD - GET KEYBOARD STATUS
  AH = 01h
Return: ZF set if no keystroke available
  ZF clear if keystrokes in buffer
      AX = next keystroke (AH=00h for ASCII keys -- no scan code)
SeeAlso: AH=00h,AH=02h,AH=03h,AH=04h,AH=05h,INT 16/AH=01h
--------b-4A02--------------------------------
INT 4A - TI Professional PC - KEYBOARD - GET KEYBOARD MODE
  AH = 02h
Return: AL = shift states (see #03213)
SeeAlso: AH=00h,AH=02h,AH=03h,AH=04h,AH=05h,INT 16/AH=02h
```

```
Bitfields for TI Professional PC keyboard shift states:
Bit(s)  Description (Table 03213)
 0  Ctrl key pressed
 1  Alt key pressed
 2  either Shift key pressed
 3-6  0
 7  CapsLock is ON
--------b-4A03-------------------------------
INT 4A - TI Professional PC - KEYBOARD - FLUSH KEYBOARD BUFFER
  AH = 03h
Return: nothing
SeeAlso: AH=00h,AH=02h,AH=03h,AH=04h,AH=05h
--------b-4A04-------------------------------
INT 4A - TI Professional PC - KEYBOARD - SEND COMMAND TO KEYBOARD
  AH = 04h
  AL = command
       00h reset to default states
       01h enable auto-repeat (default)
       02h disable auto-repeat
       03h lock keyboard
       04h unlock keyboard (default)
       05h enable keyclick (requires hardware modification to work)
       06h disable keyclick (default)
Return: nothing
SeeAlso: AH=00h,AH=02h,AH=03h,AH=04h,AH=05h
--------b-4A05-------------------------------
INT 4A - TI Professional PC - KEYBOARD - INSERT CHARACTER INTO KEYBOARD BUFFER
  AH = 05h
  BX = character code (BH=00h if ASCII character, BL=00h/BH nonzero for
         extended codes) (see #03214)
Return: ZF set if keyboard buffer was already full
  ZF clear if keystroke inserted into buffer
SeeAlso: AH=00h,AH=02h,AH=03h,AH=04h,AH=05h,INT 5B"TI"

(Table 03214)
Values for TI Professional PC scan/character codes:
 Scan Key Normal  Shift Ctrl  Alt Notes
 00h    -- unused
 01h  F5  3F00h 5800h 6200h 6C00h
 02h  F6  4000h 5900h 6300h 6D00h
 03h  F7  4100h 5A00h 6400h 6E00h
```

```
04h  F8  4200h 5B00h 6500h 6F00h
05h  F9  4300h 5C00h 6600h 7000h
06h  F10 4400h 5D00h 6700h 7100h
07h  F11 4500h 0800h 0A00h 0C00h
08h  F12 4600h 0900h 0B00h 0D00h
09h  1 ! 0031h 0021h ----  7800h
0Ah  2 @ 0032h 0040h 0300h 7900h
0Bh  3 # 0033h 0023h ----  7A00h
0Ch  4 $ 0034h 0024h ----  7B00h
0Dh  5 % 0035h 0025h ----  7C00h
0Eh  6 ^ 0036h 005Eh 001Eh 7D00h
0Fh  7 & 0037h 0026h ----  7E00h
10h  8 * 0038h 002Ah ----  7F00h
11h  9 ( 0039h 0028h ----  8000h
12h  0 ) 0030h 0029h ----  8100h
13h  - _ 002Dh 005Fh 001Fh 8200h
14h  = + 003Dh 002Bh ----  8300h
15h BACK SPACE 0008h 0008h 007Fh ----
16h  ` ~ 0060h 007Eh ----  ----
17h  NUM = 003Dh 003Dh 003Dh 8C00h
18h  NUM + 002Bh 002Bh 002Bh 8D00h
19h   NUM SPAC 0020h 0020h 0020h 8E00h
1Ah   NUM TAB  0009h 0F00h 0009h 8F00h
1Bh  NUM 1 0031h 0031h 0031h (alt-###) [Note 5]
1Ch  (unused)
1Dh  NUM 0 0030h 0030h 0030h (alt-###) [Note 5]
1Eh  NUM ENTER 000Dh 000Dh 000Dh ----
1Fh  NUM 4 0034h 0034h 0034h (alt-###) [Note 5]
20h  NUM 5 0035h 0035h 0035h (alt-###) [Note 5]
21h  NUM 9 0039h 0039h 0039h (alt-###) [Note 5]
22h  NUM - 002Dh 002Dh 002Dh ----
23h  NUM 2 0032h 0032h 0032h (alt-###) [Note 5]
24h-26h  -- unused
27h  NUM 7 0037h 0037h 0037h (alt-###) [Note 5]
28h  NUM 8 0038h 0038h 0038h (alt-###) [Note 5]
29h  NUM 6 0036h 0036h 0036h (alt-###) [Note 5]
2Ah  NUM , 002Ch 002Ch 002Ch ----
2Bh  NUM 3 0033h 0033h 0033h (alt-###) [Note 5]
2Ch  NUM . 002Eh 002Eh 002Eh ----
2Dh  PRINT 7200h [Note2] ----  ----  [Notes 1,2]
2Eh   RtArrow  4D00h 8A00h 7400h 4E00h
```

```
2Fh  INS 5200h 2800h 2900h 2A00h [Note 1]
30h  DEL 5300h 3800h 3900h 3A00h [Note 1]
31h  TAB 0009h 0F00h 0009h ----
32h  Q 0071h 0051h 0011h 1000h
33h  W 0077h 0057h 0017h 1100h
34h  E 0065h 0045h 0005h 1200h
35h  R 0072h 0052h 0012h 1300h
36h  T 0074h 0054h 0014h 1400h
37h  Y 0079h 0059h 0019h 1500h
38h  U 0075h 0055h 0015h 1600h
39h  I 0069h 0049h 0009h 1700h
3Ah  O 006Fh 004Fh 000Fh 1800h
3Bh  P 0070h 0050h 0010h 1900h
3Ch  [ { 005Bh 007Bh 001Bh ----
3Dh  ] } 005Dh 007Dh 001Dh ----
3Eh  LINE FEED 000Ah 000Ah 7500h 4F00h
3Fh  BRK/PAUS  [Note3] [Note4] ----  ----  [Notes 1,3,4]
40h  UpArrow 4800h 8800h 8400h 4900h
41h  ESC 001Bh 001Bh 001Bh ----
42h  A 0061h 0041h 0001h 1E00h
43h  S 0073h 0053h 0013h 1F00h
44h  D 0064h 0044h 0004h 2000h
45h  F 0066h 0046h 0006h 2100h
46h  G 0067h 0047h 0007h 2200h
47h  H 0068h 0048h 0008h 2300h
48h  J 006Ah 004Ah 000Ah 2400h
49h  K 006Bh 004Bh 000Bh 2500h
4Ah  L 006Ch 004Ch 000Ch 2600h
4Bh  ; : 003Bh 003Ah ----  ----
4Ch  ' " 0027h 0022h ----  ----
4Dh  RETURN  000Dh 000Dh 000Dh ----
4Eh  \ | 005Ch 007Ch 001Ch ----
4Fh  LeftArrow 4B00h 8B00h 7300h 4C00h
50h  HOME  4700h 8600h 7700h 8500h
51h  Space Bar 0020h 0020h 0020h 0020h
52h  Z 007Ah 005Ah 001Ah 2C00h
53h  X 0078h 0058h 0018h 2D00h
54h  C 0063h 0043h 0003h 2E00h
55h  V 0076h 0056h 0016h 2F00h
56h  B 0062h 0042h 0002h 3000h
57h  N 006Eh 004Eh 000Eh 3100h
```

```
 58h  M 006Dh 004Dh 000Dh 3200h
 59h  , < 002Ch 003Ch ----  ----
 5Ah  PRINT 7200h [Note2] ----  ----  [Notes 1,2]
 5Bh  . > 002Eh 003Eh ----  ----
 5Ch  / ? 002Fh 003Fh ----  ----
 5Dh  (unused)
 5Eh  DEL 5300h 3800h 3900h 3A00h [Note 1]
 5Fh  INS 5200h 2800h 2900h 2A00h [Note 1]
 60h  DownArrow 5000h 8900h 7600h 5100h
 61h-63h  -- unused
 64h  BRK/PAUS  [Note3] [Note4] ----  ----  [Notes 1,3,4]
 65h  F1  3B00h 5400h 5E00h 6800h
 66h  F2  3C00h 5500h 5F00h 6900h
 67h  F3  3D00h 5600h 6000h 6A00h
 68h  F4  3E00h 5700h 6100h 6B00h
 69h-6Fh  -- unused
Notes:  [1] four of the keys can have differing scan codes, depending on the
    actual keyboard; the BIOS accepts either scan code ("normal": 2Fh,
    30h, 5Ah, 64h; "alternate": 2Dh,3Fh,5Eh,5Fh) for any of these keys
  [2] Shift-Print invokes INT 5E for a screen dump; the PRTSCRN.DEV
    device driver also supports Alt-Print, Ctrl-Print, Shift-Alt-Print,
    and Shift-Ctrl-Print for dumping graphics in various permutations
  [3] BRK/PAUS invokes INT 5C for a pause, then stuffs 0100h into the
    keyboard buffer
  [4] Shift-BRK/PAUS invokes INT 5D for the Break, then stuffs 0000h
    into the keyboard buffer; MS-DOS hooks INT 5D to keep the 0000h from
    appearing in the keyboard buffer
  [5] on the TI Pro, one enters an arbitrary character slightly
    differently than on a standard PC: exactly three numberpad digits
    must be pressed (using leading zeros for codes less than 100), and
    the key for the requested code is inserted into the keyboard buffer
    immediately on pressing the third key.  The Alt key may be released
    and re-pressed arbitrarily often between digits without affecting
    the Alt-digit-digit-digit sequence.
  scan codes with bit 7 set are not key releases, but rather
    auto-repeated keystrokes, which the BIOS only places into the
    keyboard buffer if the buffer is empty at the time (thus avoiding
    typeahead of repeated keystrokes faster than they can be processed)
SeeAlso: #00006 at INT 09
--------h-4B---------------------------------
INT 4B - Z100 - Slave 8259 - S100 vectored line 3
```

```
SeeAlso: INT 4A"Z100",INT 4C"Z100"
--------d-4B---------------------------------
INT 4B - Common Access Method SCSI interface (draft revision 1.9)
  ES:DI -> CAM Control Block (see #03229 at INT 4F/AX=8100h)
InstallCheck: test for the string "SCSI_CAM" eight bytes past the INT 4Bh
    handler
Notes:  the CAM committee moved the interface to INT 4F after revision 1.9
    to avoid conflicting with the IBM SCSI interface and the Virtual
    DMA specification
  the only driver to date reported to use the CAM interface on INT 4B
    instead of INT 4F is from Future Domain (which has drivers for CAM
    on either interrupt)
SeeAlso: INT 4F/AX=8100h
Index:  installation check;Common Access Method SCSI interface
--------b-4B---------------------------------
INT 4B - Tandy 2000 - EQUIPMENT DETERMINATION
Return: AX = BIOS equipment list word (see #03215)
Note: this interrupt is identical to INT 11 on the Tandy 2000
SeeAlso: INT 11"EQUIPMENT",INT 4A"Tandy 2000",INT 4C"Tandy 2000"

Bitfields for Tandy 2000 BIOS equipment list:
Bit(s)  Description (Table 03215)
 0  reserved
 1  monochrome graphics installed
 2  graphics with color option installed
 3  floppy disk drive 1 installed
 4  floppy disk drive 2 installed
 5  hard disk drive 1 installed
 6  hard disk drive 2 installed
 7  unused
 8  black and white monitor
 9  color monitor
 12-10  reserved
 13 printer installed
 14 reserved
 15 unused
SeeAlso: #00226 at INT 11
--------O-4B---------------------------------
INT 4B - Acorn BBC Master 512 - "OSBYTE" - MISC FUNCTIONS USING REGISTER PARAMS
  AL = function code
  BL = first parameter
```

```
  BH = second parameter (if needed)
Return: BL = first return parameter
  BH = second return parameter
  CF depends on function
SeeAlso: INT 40"Acorn",INT 4A"Acorn",INT 4C"Acorn"
--------b-4B00-------------------------------
INT 4B - TI Professional PC - PARALLEL PORT - OUTPUT CHARACTER
  AH = 00h
  DL = printer number (00h)
  AL = character to print
Return: AH = printer status (see #03216)
Note: on the TI Pro, the BIOS only supports DL=00h; MS-DOS versions for the
    TI hook INT 4B and handle requests for DL<>00h
SeeAlso: AH=01h,AH=02h,INT 17/AH=00h
SeeAlso: INT 40"TI Professional",INT 48/AH=00h"TI Professional"
SeeAlso: INT 49/AH=01h"TI",INT 4C"TI Professional",INT 4D/AH=00h

Bitfields for TI Professional PC printer status:
Bit(s)  Description (Table 03216)
 0  timeout (function 00h only)
 3-1  unused
 4  busy
 5  paper out
 6  on-line (selected)
 7  fault
--------b-4B01-------------------------------
INT 4B - TI Professional PC - PARALLEL PORT - INITIALIZE PRINTER
  AH = 01h
  DL = printer number (00h)
Return: AH = printer status (see #03216)
Note: on the TI Pro, the BIOS only supports DL=00h; MS-DOS versions for the
    TI hook INT 4B and handle requests for DL<>00h
SeeAlso: AH=00h,AH=02h,INT 17/AH=01h
--------b-4B02-------------------------------
INT 4B - TI Professional PC - PARALLEL PORT - GET PRINTER STATUS
  AH = 02h
  DL = printer number (00h)
Return: AH = printer status (see #03216)
Note: on the TI Pro, the BIOS only supports DL=00h; MS-DOS versions for the
    TI hook INT 4B and handle requests for DL<>00h
SeeAlso: AH=00h,AH=01h,INT 17/AH=02h
```

```
--------d-4B80--------------------------------
INT 4B - IBM SCSI interface
  AH = 80h
  AL = 00h-10h (Corel PowerSCSI INT4BCAM.SYS)
  further details not yet available
--------d-4B8102DX0000----------------------
INT 4B - Virtual DMA Specification (VDS) - GET VERSION
  AX = 8102h
  DX = 0000h
Return: CF clear if successful
      AH = major version number
      AL = minor version number
      BX = product number (see #03217)
      CX = product revision number
    always 0000h for QMAPS and HPMM.SYS
    always 0001h for Microsoft's EMM386.EXE v4.20-4.41
      DX = flags (see #03219)
      SI:DI = maximum DMA buffer size
  CF set on error
      AL = error code (see #03218)
Note: bit 5 of 0040h:007Bh is supposed to be set if VDS is supported; this is
    apparently not always the case
SeeAlso: INT 2C/AX=002Bh,INT 31/AX=0400h,MEM 0040h:007Bh"4Bh"
Index:  installation check;Virtual DMA Specification

(Table 03217)
Values for VDS product number:
 0000h  for Quadtel's QMAPS and Hewlett-Packard's HPMM.SYS
 0001h  for Microsoft's EMM386.EXE
 0003h  for Windows 3.x WIN386.EXE
 0300h  OS/2 (all versions to date)
 0EDCh  for DR DOS 6.0 EMM386.SYS
 4560h  ("E`") for Qualitas' 386MAX
 4D43h  ("MC") for V Communications' Memory Commander
 5145h  ("QE") for Quarterdeck's QEMM-386
 524Dh  ("RM") for Helix's Netroom RM386

(Table 03218)
Values for VDS error code:
 01h  region not in contiguous memory
 02h  region crossed a physical alignment boundary
```

```
 03h  unable to lock pages
 04h  no buffer available
 05h  region too large for buffer
 06h  buffer currently in use
 07h  invalid memory region
 08h  region was not locked
 09h  number of physical pages greater than table length
 0Ah  invalid buffer ID
 0Bh  copy out of buffer range
 0Ch  invalid DMA channel number
 0Dh  disable count overflow
 0Eh  disable count underflow
 0Fh  function not supported
 10h  reserved flag bits set in DX

Bitfields for VDS flags:
Bit(s)  Description (Table 03219)
 0  PC/XT bus (DMA in first megabyte only)
 1  physical buffer/remap region in first megabyte
 2  automatic remap enabled
 3  all memory is physically contiguous
 4-15 reserved (zero)
--------d-4B8103-----------------------------
INT 4B - Virtual DMA Specification - LOCK DMA REGION
  AX = 8103h
  DX = flags (see #03220)
  ES:DI -> DMA descriptor structure (see #03221,#03222,#03223)
Return: CF clear if successful
      DDS physical address field filled in
      DDS buffer ID field filled (0000h if no buffer allocated)
  CF set on error
      AL = error code (see #03218)
      DDS region size field filled wth maximum contiguous length in bytes
BUGS: Windows 3.0 does not correctly support automatic remapping or copying
    in enhanced mode
  Windows 3.0 in enhanced mode does not return a correct code on error
SeeAlso: AX=8104h,AX=8105h

Bitfields for VDS flags:
Bit(s)  Description (Table 03220)
 0  reserved (zero)
```

```
 1  data should be copied into buffer (ignored if 2 set)
 2  buffer should not be allocated if region noncontiguous or crosses
    physical alignment boundary specified by 4-5
 3  don't attempt automatic remap
 4  region must not cross 64K physical alignment boundary
 5  region must not cross 128K physical alignment boundary
 6-15 reserved (zero)

Format of DMA descriptor structure (DDS):
Offset  Size  Description (Table 03221)
 00h  DWORD region size
 04h  DWORD offset
 08h  WORD  segment/selector
 0Ah  WORD  buffer ID
 0Ch  DWORD physical address

Format of Extended DMA descriptor structure (EDDS):
Offset  Size  Description (Table 03222)
 00h  DWORD region size
 04h  DWORD offset
 08h  WORD  segment/selector
 0Ah  WORD  reserved
 0Ch  WORD  number available
 0Eh  WORD  number used
 10h  DWORD region 0 physical address
 14h  DWORD region 0 size in bytes
 18h  DWORD region 1 physical address
 1Ch  DWORD region 1 size in bytes
  ...

Format of Extended DMA descriptor structure (EDDS) with page table entries:
Offset  Size  Description (Table 03223)
 00h  DWORD region size
 04h  DWORD offset
 08h  WORD  segment/selector
 0Ah  WORD  reserved
 0Ch  WORD  number available
 0Eh  WORD  number used
 10h  DWORD page table entry 0 (same as 80386 page table entry)
 14h  DWORD page table entry 1
  ...
```

```
Note: bits 1-11 of the page table entries should be zero; bit 0 set if page
    is present and locked
--------d-4B8104-----------------------------
INT 4B - Virtual DMA Specification - UNLOCK DMA REGION
  AX = 8104h
  DX = flags
       bit 0: reserved (zero)
       bit 1: data should be copied out of buffer
       bits 2-15 reserved (zero)
  ES:DI -> DMA descriptor structure (see #03221,#03222) with region size,
       physical address, and buffer ID fields set
Return: CF clear if successful
       DDS physical address field set
       DDS buffer ID field set (0000h if no buffer allocated)
  CF set on error
       AL = error code (see #03218)
       DDS region size field filled wth maximum contiguous length in bytes
Note: Windows 3.0 does not check whether the region extends beyond the end of
    a segment
BUG:  Windows 3.0 in enhanced mode does not return a correct code on error
SeeAlso: AX=8103h,AX=8106h
--------d-4B8105-----------------------------
INT 4B - Virtual DMA Specification - SCATTER/GATHER LOCK REGION
  AX = 8105h
  DX = flags (see #03224)
  ES:DI -> Extended DMA descriptor structure (see #03222,#03223)
       region size, linear segment, linear offset, and number avail
       fields set
Return: CF clear if successful
       EDDS number used field set
       if DX bit 6 set, lower 12 bits of BX = offset in first page
  CF set on error
       AL = error code (see #03218)
       EDDS region size field filled with max length in bytes that can be
       locked and described in the EDDS table
BUG:  Windows 3.0 in enhanced mode may return zero instead of the physical
    page address for pages which were originally not present
SeeAlso: AX=8103h,AX=8106h

Bitfields for VDS flags:
Bit(s)  Description (Table 03224)
```

```
 0-5  reserved (zero)
 6  EDDS should be returned with page table entries
 7  only present pages should be locked (not-present pages receive entry
    of 0000h)
 8-15 reserved (zero)
--------d-4B8106-----------------------------
INT 4B - Virtual DMA Specification - SCATTER/GATHER UNLOCK REGION
  AX = 8106h
  DX = flags (see #03225)
  ES:DI -> Extended DMA descriptor structure (see #03222,#03223) returned
       by AX=8105h
Return: CF clear if successful
  CF set on error
       AL = error code (see #03218)
Note: according to the Microsoft version of the VDS specification, the
    actual scatter/gather list is ignored, while according to the IBM
    version of the specification, "the result of a LOCK operation"
    must be provided to this function
SeeAlso: AX=8104h,AX=8105h

Bitfields for VDS flags:
Bit(s)  Description (Table 03225)
 0-5  reserved (zero)
 6  EDDS contains page table entries
 7  EDDS may contain not-present pages (entry = 0000h)
 8-15 reserved (zero)
--------d-4B8107-----------------------------
INT 4B - Virtual DMA Specification - REQUEST DMA BUFFER
  AX = 8107h
  DX = flags
       bit 0: reserved (zero)
       bit 1: data should be copied into buffer
       bits  2-15 reserved (zero)
  ES:DI -> DMA descriptor structure (see #03221) with region size set
       (also region offset and region segment if DX bit 1 set)
Return: CF clear if successful
       DDS physical address and buffer ID set
       DDS region size filled with length of buffer
  CF set on error
       AL = error code (see #03218)
SeeAlso: AX=8108h
```

```
--------d-4B8108-----------------------------
INT 4B - Virtual DMA Specification - RELEASE DMA BUFFFER
  AX = 8108h
  DX = flags
       bit 0: reserved (zero)
       bit 1: data should be copied out of buffer
       bits 2-15 reserved (zero)
  ES:DI -> DMA descriptor structure (see #03221,#03222) with buffer ID set
       (also region size/region offset/segment if DX bit 1 set)
Return: CF clear if successful
  CF set on error
       AL = error code (see #03218)
BUG:  under Windows 3.0 Enhanced mode, you must specify that data be copied
    for this function to work correctly
SeeAlso: AX=8107h
--------d-4B8109DX0000------------------------
INT 4B - Virtual DMA Specification - COPY INTO DMA BUFFER
  AX = 8109h
  DX = 0000h
  ES:DI -> DMA descriptor structure (see #03221,#03222) with buffer ID,
       region segment/offset, and region size fields set
  BX:CX = starting offset into DMA buffer
Return: CF clear if successful
  CF set on error
       AL = error code (see #03218)
BUG:  Windows 3.0 Enhanced mode does not correctly interpret the copy count
SeeAlso: AX=810Ah
--------d-4B810ADX0000------------------------
INT 4B - Virtual DMA Specification - COPY OUT OF DMA BUFFER
  AX = 810Ah
  DX = 0000h
  ES:DI -> DMA descriptor structure (see #03221,#03223) with buffer ID,
       region segment/offset, and region size fields set
  BX:CX = starting offset into DMA buffer
Return: CF clear if successful
  CF set on error
       AL = error code (see #03218)
BUG:  Windows 3.0 Enhanced mode does not correctly interpret the copy count
SeeAlso: AX=8109h
--------d-4B810B-----------------------------
INT 4B - Virtual DMA Specification - DISABLE DMA TRANSLATION
```

```
  AX = 810Bh
  BX = DMA channel number
  DX = 0000h
Return: CF clear if successful
  CF set on error
      AL = error code (see #03218)
SeeAlso: AX=810Ch
--------d-4B810C-----------------------------
INT 4B - Virtual DMA Specification - ENABLE DMA TRANSLATION
  AX = 810Ch
  BX = DMA channel number
  DX = 0000h
Return: CF clear if successful
      ZF set if disable count decremented to zero
  CF set on error
      AL = error code (see #03218)
SeeAlso: AX=810Bh
--------Q-4B810D-----------------------------
INT 4B - QEMM-386 - BUG
  AX = 810Dh
Note: the code in QEMM v5.11 and 6.00 jumps to an invalid location on this
    call
--------h-4C---------------------------------
INT 4C - Z100 - Slave 8259 - S100 vectored line 4
SeeAlso: INT 4B"Z100",INT 4D"Z100"
--------b-4C---------------------------------
INT 4C - TI Professional PC - CLOCK/ANALOG INTERFACE
  no details available
SeeAlso: INT 40"TI Professional",INT 49/AH=01h"TI"
SeeAlso: INT 4A/AH=00h"TI",INT 4B"TI Professional",INT 4D/AH=00h
SeeAlso: INT 58"TI Professional"
--------b-4C---------------------------------
INT 4C - Tandy 2000 - GET MEMORY SIZE
Return: AX = kilobytes of contiguous memory starting at 0
Note: this interrupt is identical to INT 12 on the Tandy 2000
SeeAlso: INT 12"BIOS",INT 4A"Tandy 2000",INT 4B"Tandy 2000",INT 51"Tandy 2000"
--------O-4C---------------------------------
INT 4C - Acorn BBC Master 512 - "OSCLI" - INTERPRET COMMAND LINE
  DS:BX -> CR-terminated command string
Return: FLAGS destroyed
SeeAlso: INT 40"Acorn",INT 4A"Acorn",INT 4B"Acorn"
```

```
--------h-4D---------------------------------
INT 4D - Z100 - Slave 8259 - S100 vectored line 5
SeeAlso: INT 4C"Z100",INT 4E"Z100"
--------s-4D---------------------------------
INT 4D - IBM - M-Audio Adapter SUPPORT
  no details available; supposedly documented in IBM form G571-0203-01
--------B-4D00---------------------------------
INT 4D - TI Professional PC - DISK - RESET DISK SYSTEM
  AH = 00h
  DL = drive (if bit 7 is set both hard disks and floppy disks reset)
Return: AH = status (see #00234 at INT 13/AH=01h)
  CF clear if successful (returned AH=00h)
  CF set on error
Note: this function is the same as INT 13/AH=00h on a standard PC BIOS
SeeAlso: AH=01h,AH=02h,AH=08h,AH=0Bh,INT 13/AH=00h,INT 46"TI Professional"
SeeAlso: INT 48/AH=00h"TI Professional",INT 4A/AH=00h"TI"
--------B-4D01---------------------------------
INT 4D - TI Professional PC - DISK - GET STATUS OF LAST OPERATION
  AH = 01h
  DL = drive (bit 7 set for hard disk)
Return: CF clear if status unchanged
  CF set if status changed since last call
  AH = 00h
  AL = status of previous operation (see #00234 at INT 13/AH=01h)
Notes:  this function is nearly the same as INT 13/AH=01h on a standard PC BIOS
  the TI's BIOS tranparently performs a number of retries, and an error
    status is only reported if all of the retries fail.  To get the error
    status if the operation succeeded on a retry, use AH=07h instead
SeeAlso: AH=00h,AH=07h,INT 13/AH=01h
--------B-4D02---------------------------------
INT 4D - TI Professional PC - DISK - READ SECTOR(S) INTO MEMORY
  AH = 02h
  AL = number of sectors to read (must be nonzero)
  CH = low eight bits of cylinder number
  CL = sector number 1-63 (bits 0-5)
       high two bits of cylinder (bits 6-7, hard disk only)
  DH = head number
  DL = drive number (bit 7 set for hard disk)
  ES:BX -> data buffer
Return: CF set on error
      if AH = 11h (corrected ECC error), AL = burst length
```

```
  CF clear if successful
  AH = status (see #00234 at INT 13/AH=01h)
  AL = number of sectors transferred
  ES:BX -> buffer for last sector processed (including one with errors)
SeeAlso: AH=00h,AH=01h,AH=03h,AH=04h,INT 13/AH=02h
--------B-4D03-----------------------------
INT 4D - TI Professional PC - DISK - WRITE SECTOR(S) FROM MEMORY
  AH = 03h
  AL = number of sectors to write (must be nonzero)
  CH = low eight bits of cylinder number
  CL = sector number 1-63 (bits 0-5)
       high two bits of cylinder (bits 6-7, hard disk only)
  DH = head number
  DL = drive number (bit 7 set for hard disk)
  ES:BX -> buffer containing data
Return: CF set on error
      if AH = 11h (corrected ECC error), AL = burst length
  CF clear if successful
  AH = status (see #00234 at INT 13/AH=01h)
  AL = number of sectors transferred
  ES:BX -> buffer for last sector processed (including one with errors)
SeeAlso: AH=00h,AH=01h,AH=02h,AH=04h,INT 13/AH=03h
--------B-4D04-----------------------------
INT 4D - TI Professional PC - DISK - VERIFY DISK SECTOR CRC(S)
  AH = 04h
  AL = number of sectors to verify (must be nonzero)
  CH = low eight bits of cylinder number
  CL = sector number 1-63 (bits 0-5)
       high two bits of cylinder (bits 6-7, hard disk only)
  DH = head number
  DL = drive number (bit 7 set for hard disk)
  ES:BX -> data buffer
Return: CF set on error
      if AH = 11h (corrected ECC error), AL = burst length
  CF clear if successful
  AH = status (see #00234 at INT 13/AH=01h)
  AL = number of sectors transferred
  ES:BX -> buffer for last sector processed (including one with errors)
Note: even though no data is transferred, ES:BX must still be valid
SeeAlso: AH=00h,AH=01h,AH=02h,AH=06h,INT 13/AH=04h
--------B-4D05-----------------------------
```

```
INT 4D - TI Professional PC - DISK - NOP
  AH = 05h
Note: on the TI Pro, FORMAT.COM contains direct port I/O commands to perform
    disk formatting, rather than using the BIOS
--------B-4D06--------------------------------
INT 4D - TI Professional PC - DISK - VERIFY DISK SECTOR(S)
  AH = 06h
  AL = number of sectors to verify (must be nonzero)
  CH = low eight bits of cylinder number
  CL = sector number 1-63 (bits 0-5)
       high two bits of cylinder (bits 6-7, hard disk only)
  DH = head number
  DL = drive number (bit 7 set for hard disk)
  ES:BX -> data buffer
Return: CF set on error
      if AH = 11h (corrected ECC error), AL = burst length
  CF clear if successful
  AH = status (see #00234 at INT 13/AH=01h)
  AL = number of sectors transferred
  ES:BX -> buffer for last sector processed (including one with errors)
Note: even though no data is transferred, ES:BX must still be valid because
    an actual comparison with disk data is performed, not just the CRC
    check of the standard PC BIOS or INT 4D/AH=04h
SeeAlso: AH=00h,AH=01h,AH=02h,AH=04h,INT 13/AH=04h
--------B-4D07--------------------------------
INT 4D - TI Professional PC - DISK - GET RETRY STATUS OF LAST OPERATION
  AH = 07h
  DL = drive (bit 7 set for hard disk)
Return: CF clear if status unchanged
  CF set if status changed since last call
  AH = 00h
  AL = status of previous operation (see #00234 at INT 13/AH=01h)
Notes:  this function is nearly the same as INT 13/AH=01h on a standard PC BIOS
  the TI's BIOS tranparently performs a number of retries; this function
    returns the error status of a failed operation even if the operation
    succeeded on a retry
SeeAlso: AH=00h,AH=01h,INT 13/AH=01h
--------B-4D08--------------------------------
INT 4D - TI Professional PC - DISK - SET STANDARD DEVICE INTERFACE TABLE
  AH = 08h
  DL = drive number (00h-03h)
```

```
  AL = drive type
       00h single-sided 48 tpi (40-track, 8 sectors, 512 bytes/sector)
       01h double-sided 48 tpi (40-track, 8 sectors, 512 bytes/sector)
       02h single-sided 96 tpi (80-track, 8 sectors, 512 bytes/sector)
       03h double-sided 96 tpi (80-track, 8 sectors, 512 bytes/sector)
Return: nothing???
SeeAlso: AH=00h,AH=09h
--------B-4D09-------------------------------
INT 4D - TI Professional PC - DISK - SET DEVICE INTERFACE TABLE ADDRESS
  AH = 09h
  DL = drive number (00h-07h)
  ES:BX -> Device Interface Table (see #03226)
Return: nothing???
SeeAlso: AH=00h,AH=08h,AH=0Ah,INT 1E

Format of TI Professional PC Device Interface Table:
Offset  Size  Description (Table 03226)
 00h  DWORD -> entry point for disk routine
 04h  WORD  bytes per sector
 06h  BYTE  sectors per track
 07h  BYTE  number of heads
 08h  BYTE  number of cylinders
 09h  BYTE  retry count
 0Ah  BYTE  precompensation start
SeeAlso: #01264 at INT 1E
--------B-4D0A-------------------------------
INT 4D - TI Professional PC - DISK - GET DEVICE INTERFACE TABLE ADDRESS
  AH = 0Ah
  DL = drive number (00h-07h)
Return: AH = status
  ES:BX -> Device Interface Table (see #03226)
SeeAlso: AH=00h,AH=08h,AH=09h,INT 1E
--------B-4D0B-------------------------------
INT 4D - TI Professional PC - DISK - TURN OFF ALL DRIVES
  AH = 0Bh
Return: AH = 00h
Note: used for diagnostics or to conserve power
SeeAlso: AH=00h
--------h-4E-------------------------------
INT 4E - Z100 - Slave 8259 - S100 vectored line 6
SeeAlso: INT 4D"Z100",INT 4F"Z100"
```

```
--------b-4E00--------------------------------
INT 4E - TI Professional PC - TIME-OF-DAY CLOCK - SET BIOS DATE
  AH = 00h
  BX = number of days since January 1, 1980
Return: nothing
SeeAlso: AH=01h,AH=02h
SeeAlso: INT 40"TI Professional",INT 48/AH=00h"TI Professional"
SeeAlso: INT 4A/AH=00h"TI",INT 4F"TI Professional"
--------b-4E01--------------------------------
INT 4E - TI Professional PC - TIME-OF-DAY CLOCK - SET BIOS TIME
  AH = 01h
  CH = hours
  CL = minutes
  DH = seconds
  DL = hundredths
Return: nothing
Note: the BIOS does not validate the data passed to this function
SeeAlso: AH=00h,AH=02h
--------b-4E02--------------------------------
INT 4E - TI Professional PC - TIME-OF-DAY CLOCK - GET BIOS DATA AND TIME
  AH = 02h
Return: AX = number of days since January 1, 1980
  CH = hours
  CL = minutes
  DH = seconds
  DL = hundredths
SeeAlso: AH=00h,AH=01h
--------h-4F--------------------------------
INT 4F - Z100 - Slave 8259 - S100 vectored line 7
SeeAlso: INT 4E"Z100"
--------b-4F--------------------------------
INT 4F - TI Professional PC - SYSTEM CONFIGURATION CALL
Return: AX = system configuration word (see #03227)
  BX = size of contiguous DOS memory in paragraphs
SeeAlso: INT 11"BIOS",INT 12"BIOS",INT 40"TI Professional",INT 48/AH=09h
SeeAlso: INT 49/AH=01h"TI",INT 4B"TI Professional",INT 4D/AH=00h
SeeAlso: INT 4E"TI Professional"

Bitfields for TI Professional PC system configuration:
Bit(s)  Description (Table 03227)
 0  floppy drive 0 (A:, internal) installed
```

```
 1  floppy drive 1 (B:, internal) installed
 2  floppy drive 2 (C:, external) installed
 3  floppy drive 3 (D:, external) installed
 4  drive A: is 96tpi (80 tracks)
 5  drive A: is double-sided
 6  60 Hz power instead of 50 Hz
 7  hard disk (E: or E:/F:) installed
 8  serial port 1 installed
 9  serial port 2 installed
 10 serial port 3 installed
 11 serial port 4 installed
 14-12  installed graphics RAM
  000 none (text-only system)
  001 bank A only (graphics limited to 2 of 8 colors)
  111 banks A/B/C (graphics supports 8 of 8 colors)
 15 clock/analog board installed
--------d-4F0081-----------------------------
INT 4F - ATA Software Programming Interface (ATASPI) - SEND ATASPI REQUEST
  AX = 0081h
  ES:BX -> ATA Request Block (see #90002)
Return: AH = 00h always???
Note: this interface has probably appeared in some later revisions of ATASPI
    than 0.72
SeeAlso: INT 21/AX=4402h"ATASPI",INT 4F/AX=0082h
--------d-4F0082CX8765-----------------------
INT 4F - ATA Software Programming Interface (ATASPI) - INSTALLATION CHECK
  AX = 0082h
  CX = 8765h
  DX = CBA9h
Return: AH = 00h if installed
      CX = 9ABCh
      DX = 5678h
      ES:DI -> "$ATAMGR$"
Note: this interface has probably appeared in some later revisions of ATASPI
    than 0.72
SeeAlso: INT 21/AX=4402h"ATASPI",INT 4F/AX=0081h
--------d-4F8100-----------------------------
INT 4F - Common Access Method SCSI interface rev 2.3 - SEND CCB TO XPT/SIM
  AX = 8100h
  ES:BX -> CAM Control Block (CCB) (see #03229)
Return: AH = status
```

```
      00h successful
      01h invalid CCB address (0000h:0000h)
Note: the SCSI Interface Module (SIM) may complete the requested function
    and invoke the completion callback function before this call returns
SeeAlso: AX=8200h,INT 2F/AX=7F01h,INT 4B"Common Access Method"

(Table 03228)
Values for CAM function code:
 00h  NOP
 01h  execute SCSI I/O
 02h  get device type
 03h  path inquiry
 04h  release SIM queue
 05h  set async callback
 06h  set device type
 07h-0Fh reserved
 10h  abort SCSI command
 11h  reset SCSI bus
 12h  reset SCSI device
 13h  terminate I/O process
 14h-1Fh reserved
 20h  engine inquiry
 21h  execute engine request
 22h-2Fh reserved
 30h  enable logical unit number
 31h  execute target I/O
 32h-7Fh reserved
 80h-FFh vendor-specific functions

Format of CAM Control Block:
Offset  Size  Description (Table 03229)
 00h  DWORD physical address of this CCB
 04h  WORD  CAM control block length
 06h  BYTE  function code (see #03228)
 07h  BYTE  CAM status (see #03232)
 08h  BYTE  SCSI status
 09h  BYTE  path ID (FFh = XPT)
 0Ah  BYTE  target ID
 0Bh  BYTE  logical unit number
 0Ch  WORD  CAM flags (see #03230)
 0Eh  BYTE  CAM address flags (see #03231)
```

```
 0Fh  BYTE  target-mode flags (see #03233)
---function 02h---
 10h  DWORD pointer to 36-byte buffer for inquiry data or 0000h:0000h
 14h  BYTE  peripheral device type of target logical unit number
---function 03h---
 10h  BYTE  version number (00h-07h prior to rev 1.7, 08h = rev 1.7,
    09h-FFh = rev no, i.e. 23h = rev 2.3)
 11h  BYTE  SCSI capabilities (see #03234)
 12h  BYTE  target mode support
    bit 7: processor mode
    bit 6: phase-cognizant mode
    bit 5-0: reserved
 13h  BYTE  miscellaneous flags
    bit 7: scanned high to low instead of low to high
    bit 6: removables not included in scan
    bit 5: inquiry data not kept by XPT
    bits 4-0: reserved
 14h  WORD  engine count
 16h 14 BYTEs vendor-specific data
 24h  DWORD size of private data area
 28h  DWORD asynchronous event capabilities (see #03235)
 2Ch  BYTE  highest path ID assigned
 2Dh  BYTE  SCSI device ID of initiator
 2Eh  2 BYTEs reserved
 30h 16 BYTEs SIM vendor ID
 40h 16 BYTEs HBA (host bus adaptor) vendor ID
 50h  4 BYTEs operating-system dependant usage
---functions 00h,04h,11h,12h---
 no additional fields
---function 05h---
 10h  DWORD asynchronous event enables (refer to function 03h above)
 14h  DWORD pointer to asynchronous callback routine (see #03241)
 18h  DWORD pointer to peripheral driver buffer
 1Ch  BYTE  size of peripheral buffer
---function 06h---
 10h  BYTE  peripheral device type of target
---functions 10h,13h---
 10h  DWORD pointer to CCB to be aborted
---function 20h---
 10h  WORD  engine number
 12h  BYTE  engine type
```

```
    00h buffer memory
    01h lossless compression
    02h lossy compression
    03h encryption
 13h  BYTE  engine algorithm ID
    00h vendor-unique
    01h LZ1 variation 1 (STAC)
    02h LZ2 variation 1 (HP DCZL)
    03h LZ2 variation 2 (Infochip)
 14h  DWORD engine memory size
---function 21h---
 10h  DWORD pointer to peripheral driver
 14h  4 BYTEs reserved
 18h  DWORD OS-dependent request-mapping info
 1Ch  DWORD address of completion callback routine
 20h  DWORD pointer to scatter/gather list or data buffer
 24h  DWORD length of data transfer
 28h  DWORD pointer to engine buffer data
 2Ch  2 BYTEs reserved
 2Eh  WORD  number of scatter/gather entries
 30h  DWORD maximum destination data length
 34h  DWORD length of destination data
 38h  DWORD source residual length
 3Ch 12 BYTEs reserved
 48h  DWORD OS-dependent timeout value
 4Ch  4 BYTEs reserved
 50h  WORD  engine number
 52h  WORD  vendor-unique flags
 54h  4 BYTEs reserved
 58h  N BYTEs private data area for SIM
---function 30h---
 10h  WORD  group 6 vendor-unique CDB length
 12h  WORD  group 7 vendor-unique CDB length
 14h  DWORD pointer to target CCB list
 18h  WORD  number of target CCBs
---other functions---
 10h  DWORD pointer to peripheral driver
 14h  DWORD pointer to next CCB
 18h  DWORD OS-dependent request mapping information
 1Ch  DWORD address of completion callback routine (see #03240)
 20h  DWORD pointer to scatter/gather list or data buffer
```

```
 24h  DWORD length of data transfer
 28h  DWORD pointer to sense info buffer
 2Ch  BYTE  length of sense info buffer
 2Dh  BYTE  CDB length
 2Eh  WORD  number of scatter/gather entries
    scatter/gather list is array of 2N DWORDs, each pair specifying
      the address and length of a data block
 30h  4 BYTEs vendor-specific data
 34h  BYTE  (ret) SCSI status
 35h  BYTE  (ret) auto-sense residual length
 36h  2 BYTEs reserved
 38h  DWORD (ret) residual length
 40h 12 BYTEs Command Descriptor Block (CDB) (see #03236,#03237,#03238)
 44h  DWORD OS-dependent timeout value
 48h  DWORD pointer to message buffer
 4Ch  WORD  length of message buffer
 4Eh  WORD  vendor-unique flags
 50h  BYTE  tag queue action
 51h  3 BYTEs reserved
 54h  N BYTEs private data area for SIM

Bitfields for CAM flags:
Bit(s)  Description (Table 03230)
 0  CDB is a pointer
 1  tagged queue action enable
 2  linked CDB
 3  disable callback on completion
 4  scatter/gather
 5  disable autosense
 7-6  direction (00 reserved, 01 in, 10 out, 11 no data transfer)
 9-8  reserved
 10 engine synchronize
 11 SIM queue freeze
 12 SIM queue priority
  1 head insertion
  0 tail insertion (normal)
 13 disable synchronous transfers \ mutually
 14 initiate synchronous transfers  / exclusive
 15 disable disconnect

Bitfields for CAM address flags:
```

```
Bit(s)  Description (Table 03231)
 7  SG list/data (0 = host, 1 = engine)
 6  CDB pointer    (6-1: 0=virtual addr, 1=phys addr)
 5  SG list/data
 4  sense buffer
 3  message buffer
 2  next CCB
 1  callback on completion
 0  reserved

(Table 03232)
Values for CAM status:
 00h  request in progress
 01h  request successful
 02h  host aborted request
 03h  unable to abort request
 04h  request completed with error
 05h  CAM is busy
 06h  invalid request
 07h  invalid path ID
 08h  no such SCSI device
 09h  unable to terminate I/O process
 0Ah  timeout on target selection
 0Bh  timeout on command
 0Dh  receive message rejection
 0Eh  sent/received SCSI bus reset
 0Fh  detected uncorrectable parity error
 10h  Autosense request failed
 11h  no HBA detected
 12h  data over/underrun
 13h  bus freed unexpectedly
 14h  target bus phase sequence failure
 15h  CCB too small
 16h  requested capability not available
 17h  sent bus device reset
 18h  terminate I/O process
 38h  invalid LUN
 39h  invalid target ID
 3Ah  unimplemented function
 3Bh  nexus not established
 3Ch  invalid initiator ID
```

```
 3Dh  received SCSI Command Descriptor Block
 3Eh  LUN already enabled
 3Fh  SCSI bus busy
Note: bit 6 set to indicate frozen SIM queue
  bit 7 set to indicate valid autosense

Bitfields for CAM target-mode flags:
Bit(s)  Description (Table 03233)
 7  data buffer valid
 6  status valid
 5  message buffer valid
 4  reserved
 3  phase-cognizant mode
 2  target CCB available
 1  disable autodisconnect
 0  disable autosave/restore

Bitfields for SCSI capabilities:
Bit(s)  Description (Table 03234)
 7  modify data pointers
 6  wide bus (32 bits)
 5  wide bus (16 bits)
 4  synchronous transfers
 3  linked commands
 2  reserved
 1  tagged queueing
 0  soft reset

Bitfields for CAM asynchronous event capabilities:
Bit(s)  Description (Table 03235)
 31-24  vendor-specific
 23-8 reserved
 7  new devices found during rescan
 6  SIM module deregistered
 5  SIM module registered
 4  sent bus device reset to target
 3  SCSI AEN
 2  reserved
 1  unsolicited reselection
 0  unsolicited SCSI bus reset
```

```
Format of Six-Byte SCSI Command Descriptor Block (CDB):
Offset  Size  Description (Table 03236)
 00h  BYTE  operation code (see #03239)
 01h  BYTE  logical unit number (bits 7-5), SCSI-1/SCSI-2
    MSB of logical block address (bits 4-0)
 02h  WORD  logical block address (low word)
 04h  BYTE  transfer length
 05h  BYTE  control byte
SeeAlso: #03237,#03238

Format of Ten-Byte SCSI Command Descriptor Block (CDB):
Offset  Size  Description (Table 03237)
 00h  BYTE  operation code (see #03239)
 01h  BYTE  logical unit number (bits 7-5), SCSI-1/SCSI-2
    reserved in SCSI-3
 02h  DWORD logical block address (low word)
 06h  BYTE  reserved
 07h  WORD  transfer length
 09h  BYTE  control byte
SeeAlso: #03236,#03238

Format of Twelve-Byte SCSI Command Descriptor Block (CDB):
Offset  Size  Description (Table 03238)
 00h  BYTE  operation code (see #03239)
 01h  BYTE  logical unit number (bits 7-5), SCSI-1/SCSI-2
    reserved in SCSI-3
 02h  DWORD logical block address (low word)
 06h  DWORD transfer length
 0Ah  BYTE  reserved
 0Bh  BYTE  control byte
SeeAlso: #03236,#03237

(Table 03239)
Values for SCSI CDB operation code for direct-access devices:
 00h  Test Unit Ready
 01h  Rezero Unit
 03h  Request Sense
 04h  Format Unit
 07h  Reassign Blocks
 08h  Read (6-byte CDB)
 0Ah  Write (6-byte CDB)
```

```
 0Bh  Seek (6-byte CDB)
 12h  Inquiry
 15h  Mode Select (6-byte CDB)
 16h  Reserve
 17h  Release
 18h  Copy
 1Ah  Mode Sense (6-byte CDB)
 1Bh  Start/Stop Unit
 1Ch  Receive Diagnostic Results
 1Dh  Send Diagnostic
 1Eh  Prevent/Allow Medium Removal
 25h  Read Capacity
 28h  Read (10-byte CDB)
 2Ah  Write (10-byte CDB)
 2Bh  Seek (10-byte CDB)
 2Eh  Write and Verify
 2Fh  Verify
 30h  Search Data High
 31h  Search Data Equal
 32h  Search Data Low
 33h  Set Limits
 34h  Prefetch
 35h  Synchronize Cache
 36h  Lock/Unlock Cache
 37h  Read Defect Data
 39h  Compare
 3Ah  Copy and Verify
 3Bh  Write Buffer
 3Ch  Read Buffer
 3Eh  Read Long
 3Fh  Write Long
 40h  Change Definition
 41h  Write Same
 4Ch  Log Select
 4Dh  Log Sense
 55h  Mode Select (10-byte CDB)
 5Ah  Mode Sense (10-byte CDB)
SeeAlso: #03236,#03237,#03238

(Table 03240)
Values completion callback function is called with:
```

```
  interrupts disabled
  ES:BX -> completed CCB

(Table 03241)
Values asynchronous callback function is called with:
  AH = opcode
  AL = path ID generating callback
  DH = target ID causing event
  DL = LUN causing event
  CX = data byte count (if applicable)
  ES:BX -> data buffer (if applicable)
Return: all registers preserved
--------d-4F8200CX8765-----------------------
INT 4F - Common Access Method SCSI interface rev 2.3 - INSTALLATION CHECK
  AX = 8200h
  CX = 8765h
  DX = CBA9h
Return: AH = 00h if installed
      CX = 9ABCh
      DX = 5678h
      ES:DI -> "SCSI_CAM"
SeeAlso: AX=8100h,INT 4B"Common Access Method"
--------N-50----------------------------------
INT 50 - TIL Xpert AIM (X.25)
  AH = function
--------H-50----------------------------------
INT 50 - IRQ0 relocated by DESQview
Range:  INT 50 to INT F8, selected automatically
Notes:  this is the default location for older versions; DESQview v2.26+
    searches for unused ranges of interrupts and uses the lowest
    available range in its list for relocating these IRQs and the next
    lowest for relocating IRQ8-IRQ15
  a range of eight interrupts starting at a multiple of 8 is considered
    available if all vectors are identical and it has not been excluded
    with an /XB:nn commandline switch
  the list of ranges for v2.26 is 50h,58h,68h,78h,F8h (if < two of these
    are available, F8h and then 50h are used anyway)
  the list of ranges for v2.31+ is 68h,78h,88h-B8h,F8h (if < two of these
    are available, F8h and then F0h are used anyway)
SeeAlso: INT 08"IRQ0",INT 51"DESQview",INT 54"DESQview",INT 58"DESQview"
SeeAlso: INT D8"Screen Thief"
```

```
--------H-50---------------------------------
INT 50 - IRQ0 relocated by IBM 3278 emulation control program
SeeAlso: INT 51"IBM 3278"
--------H-50---------------------------------
INT 50 - IRQ0 relocated by OS/2 v1.x
SeeAlso: INT 51"OS/2"
----------50---------------------------------
INT 50 - TI Professional PC - FATAL SOFTWARE ERROR TRAP
Desc: the default handler generates a System Error message and halts the
    computer such that only Ctrl-Alt-Del can restart operation
Note: documented as "for system use only"; intended for multi-tasking
    software
SeeAlso: INT 40"TI Professional",INT 4F"TI Professional"
SeeAlso: INT 51"TI Professional",INT 53"TI Professional"
--------V-500000-----------------------------
INT 50 - Vanderaart TEXT WINDOWS, PC Thuis Shell - OPEN TEXT WINDOW
  AX = 0000h
  ES:BX -> name string or ES:0000h if none
  CH,CL = row,column of upper left corner
  DH,DL = row,column of lower right corner
Return: AX = window handle or
      0000h if not installed
      FFFFh on error
SeeAlso: AX=0001h,AX=0002h"TEXT WINDOWS"
--------V-500001-----------------------------
INT 50 - Vanderaart TEXT WINDOWS, PC Thuis Shell - CLOSE TEXT WINDOW
  AX = 0001h
  DI = window handle
SeeAlso: AX=0000h
--------V-500002-----------------------------
INT 50 - Vanderaart TEXT WINDOWS - PUT CHARACTER IN WINDOW
  AX = 0002h
  BL = character
  BH = attribute
  DL = column
  DH = row
  DI = window handle
Return: AX = status
      0000h if successful
      FFFFh if outside window
SeeAlso: AX=0000h
```

```
--------l-500002-----------------------------
INT 50 - PC Thuis Organizer Shell - PLOT TEXT
  AX = 0002h
  ES:BX -> text string
  DH,DL = row,column of upper left corner
  DI = window handle
Return: AX = status
      0000h successful (text fits in window)
      FFFFh error
Program: The PC Thuis Organizer Shell was written by John Vanderaart and
    published in the June/July 1990 issue of PC Thuis Power magazine
--------V-500003-----------------------------
INT 50 - Vanderaart TEXT WINDOWS - OUTPUT LINE TO WINDOW
  AX = 0003h
  ES:BX -> text string
  CX = string length (0000h if ASCIZ string)
  DL = position (FFh centered, else flush left)
  DH = starting row
  DI = window handle
Return: AX = status
      0000h successful
      FFFFh did not fit in window
--------l-500003-----------------------------
INT 50 - PC Thuis Organizer Shell - WRITE FILE
  AX = 0003h
  ES:BX -> data to be written
  CX = number of bytes to write
  DS:SI -> filename
Return: AX = status
      0000h successful
      FFFFh error
SeeAlso: AX=0004h"Shell"
--------V-500004-----------------------------
INT 50 - Vanderaart TEXT WINDOWS - GET KEY
  AX = 0004h
  CH = type
      00h any key
      01h 'J' or 'N' (Dutch for yes/no)
Return: AX = key
SeeAlso: INT 16/AH=00h
--------l-500004-----------------------------
```

```
INT 50 - PC Thuis Organizer Shell - READ FILE
  AX = 0004h
  ES:BX -> buffer for data
  CX = number of bytes to read or 0000h for entire file
  DL = file type
       01h setting shell
       02h setting sterm
       03h INT21 file
  DS:SI -> filename
Return: AX = status
       0000h successful
       FFFFh error
Note: file type numbers are maintained by John Vanderaart; if a new file type
    is needed, a type number should be requested from him through the
    magazine:
    PC Thuis BV
    Spaarne 55
    2011 CE HAARLEM
    The Netherlands
SeeAlso: AX=0003h"Shell"
--------V-500005-----------------------------
INT 50 - Vanderaart TEXT WINDOWS - CHANGE ATTRIBUTE
  AX = 0005h
  BL = new attribute
  CH,CL = row,column of upper left corner
  DH,DL = row,column of lower right corner
  DI = window handle
--------l-500005-----------------------------
INT 50 - PC Thuis Organizer Shell - PROMPT YES/NO
  AX = 0005h
  ES:BX -> prompt string (ES:0000h if no prompt)
Return: AX = key pressed
       0000h "J" (Dutch "Ja" = "Yes")
       FFFFh "N" (Dutch "Nee" = "No")
Program: The PC Thuis Organizer Shell was written by John Vanderaart and
    published in the June/July 1990 issue of PC Thuis Power magazine
SeeAlso: AX=0008h"PC Thuis"
--------V-500006-----------------------------
INT 50 - Vanderaart TEXT WINDOWS - EDIT LINE IN WINDOW
  AX = 0006h
  ES:BX -> text string
```

```
  CH = type of input (see #03242)
  DH,DL = row,column of upper left corner
  DI = window handle
Return: AX = key which terminated entry
      0000h Enter
      0001h Esc
      0002h Down arrow
      0003h Up arrow
      0004h F10

(Table 03242)
Values for type of input to Vanderaart Text Windows:
 00h  everything
 01h  uppercase only
 02h  positive numbers
 03h  Dutch postal code ("9999 AA")
 04h  'J' or 'N' (Dutch yes/no)
 05h  telephone or FAX number
 06h  positive or negative number
 07h  date (dd/mm/yy)
 08h  money
 09h  '1' through '8'
 0Ah  '1' through '4'
 0Bh  uppercase filenames
--------l-500006-----------------------------
INT 50 - PC Thuis Organizer Shell - ALERT USER
  AX = 0006h
  ES:BX -> string
--------l-500007-----------------------------
INT 50 - PC Thuis Organizer Shell - DO LINE
  AX = 0007h
  ES:BX -> text string
  CX = string length in bytes (0000h if NUL-terminated)
  DL = FFh to center string, else flush left
  DH = upper left row
  DI = window handle
Return: AX = status
      0000h successful
      FFFFh error
Program: The PC Thuis Organizer Shell was written by John Vanderaart and
    published in the June/July 1990 issue of PC Thuis Power magazine
```

```
SeeAlso: AX=0008h
--------l-500008-----------------------------
INT 50 - PC Thuis Organizer Shell - DO MENU
  AX = 0008h
  ES:BX -> menu structure
Return: AL = index 1 or FFh if not selected
  AH = index 2 or FFh if not selected
  BL = index 3 or FFh if not selected
  BH = index 4 or FFh if not selected
SeeAlso: AX=0005h"PC Thuis",AX=0007h,AX=000Ch
--------l-500009-----------------------------
INT 50 - PC Thuis Organizer Shell - MESSAGE ON
  AX = 0009h
  ES:BX -> message string
SeeAlso: AX=000Ah
--------l-50000A-----------------------------
INT 50 - PC Thuis Organizer Shell - MESSAGE OFF
  AX = 000Ah
SeeAlso: AX=0009h
--------l-50000B-----------------------------
INT 50 - PC Thuis Organizer Shell - CHANGE ATTRIBUTE
  AX = 000Bh
  BL = new attribute
  CH,CL = row,column of upper left corner
  DH,DL = row,column of lower right corner
  DI = window handle
--------l-50000C-----------------------------
INT 50 - PC Thuis Organizer Shell - DO REQUEST
  AX = 000Ch
  ES:BX -> request structure
Return: AX = status
      0000h confirmed
      FFFFh denied
SeeAlso: AX=0008h
--------l-50000D-----------------------------
INT 50 - PC Thuis Organizer Shell - EDIT LINE
  AX = 000Dh
  ES:BX -> text string
  CL = length
  CH = input type (see #03243)
  DH,DL = row,column of upper left corner
```

```
  DI = window handle
Return: AX = result code
Program: The PC Thuis Organizer Shell was written by John Vanderaart and
    published in the June/July 1990 issue of PC Thuis Power magazine

Bitfields for input type:
Bit(s)  Description (Table 03243)
 0  force uppercase
 1  integer
 2  no spaces allowed
 3  no cursor keys
--------l-50000E-----------------------------
INT 50 - PC Thuis Organizer Shell - PLOT CHARACTER
  AX = 000Eh
  BL = character
  BH = attribute
  DH,DL = row,column at which to plot
  DI = window handle
Return: AX = status
      0000h successful
      FFFFh errror
--------l-50000F-----------------------------
INT 50 - PC Thuis Organizer Shell - EMPTY WINDOW
  AX = 000Fh
  BL = character
  BH = attribute
  DI = window handle
--------l-500010-----------------------------
INT 50 - PC Thuis Organizer Shell - TRACE MENU
  AX = 0010h
  ES:BX -> first menu structure
  CL = hotkey to look up
Return: AL = index 1 or FFh if not selected
  AH = index 2 or FFh if not selected
  BL = index 3 or FFh if not selected
  BH = index 4 or FFh if not selected
Index:  hotkeys;PC Thuis Organizer Shell
--------l-500011-----------------------------
INT 50 - PC Thuis Organizer Shell - MOVE MEMORY
  AX = 0011h
  DS:SI -> source
```

```
  ES:DI -> destination
  CX = number of bytes to move (0000h = until NUL string terminator???)
SeeAlso: AX=0012h
--------l-500012-----------------------------
INT 50 - PC Thuis Organizer Shell - COMPARE MEMORY
  AX = 0012h
  DS:SI -> source
  ES:DI -> destination
  CX = number of bytes to compare (0000h=until NUL string terminator???)
Return: AX = status
      0000h same
      FFFFh different
SeeAlso: AX=0011h
--------l-500013-----------------------------
INT 50 - PC Thuis Organizer Shell - GET KEY
  AX = 0013h
  CH = type flags
      bit 0: force uppercase
      bit 1: integer
      bit 2: no spaces
Return: AX = keystroke
--------l-500014-----------------------------
INT 50 - PC Thuis Organizer Shell - SCROLL WINDOW
  AX = 0014h
  BL = direction
      06h up
      07h down
  BH = attribute
  DI = window handle
SeeAlso: INT 10/AH=06h,INT 10/AH=07h
--------l-500015-----------------------------
INT 50 - PC Thuis Organizer Shell - GET MEMORY HANDLE
  AX = 0015h
  BL = handle size
      00h 65536 bytes (64K)
      01h 65535 bytes (64K-1)
      02h 32768 bytes (32K)
      03h 32767 bytes (32K-1)
Return: AX = segment
Program: The PC Thuis Organizer Shell was written by John Vanderaart and
    published in the June/July 1990 issue of PC Thuis Power magazine
```

```
SeeAlso: INT 21/AH=48h
--------!---Section--------------------------
```