```
Interrupt List, part 16 of 18
Copyright (c) 1989-1999,2000 Ralf Brown
--------*-62---------------------------------
INT 62 - reserved for user interrupt
--------d-62---------------------------------
INT 62 - Adaptec and OMTI controllers - DRIVE 0 DATA
Notes:  this vector stores the third four bytes of the parameter table for
    hard disk 0
SeeAlso: INT 60"Adaptec",INT 61"Adaptec",INT 63"Adaptec"
--------b-62---------------------------------
INT 62 - TI Professional PC - OPTION ROM DATA AREA POINTER (NOT A VECTOR!)
Desc: the low word of this vector contains the segment of the RAM data area
    to be used by the expansion ROM at F400h:2000h, and the high word
    contains the length of the data area; this segment and size are
    both set to 0000h if no ROM is installed at F400h:2000h
SeeAlso: INT 61"TI Professional PC",INT 63"TI Professional"
SeeAlso: INT 66"TI Professional PC"
--------b-62---------------------------------
INT 62 - HP 95LX - USED BY CALCULATOR
SeeAlso: INT 60/DI=0104h"HP 95LX"
----------62---------------------------------
INT 62 - MS SQL Server/Sybase DBLIBRARY interface - ???
  AH = function (00h to 07h)
  CX = FFFEh
  DX = FFFFh
  ???
Return: ???
InstallCheck: test for the string "DBLIBRARY" two bytes past the interrupt
    handler
SeeAlso: AH=08h"SQL"
Index:  installation check;MS SQL Server|installation check;Sybase DBLIBRARY
----------62---------------------------------
INT 62 - MPAUSE - ???
  details not yet avialable
Program: MPAUSE is a program by Manfred Michael which appeared in the German
   _EGA-Handbuch_ from  m&t-Verlag
--------N-62---------------------------------
INT 62 - XFS v1.76 - FILTERED PACKET DRIVER API
Program: XFS is a shareware Network File System client by Robert Juhasz
Desc: XFS requires a packet driver to operate.  Since it does redirections,
    etc., it provides its own packet driver entry point, changing the
```

```
    signature string of the original packet driver to "XKT DRVR" so that
    it will no longer be found by the packet driver installation check
InstallCheck: scan for the signature string "PKT DRVR" three bytes past the
    interrupt handler (the same as that for the packet driver
    specification)
Range:  INT 61 to INT 66, selected by scanning for two consecutive free
    vectors and hooking the second
SeeAlso: INT 60"Packet Driver Specification",INT 61"XFS"
----------62--------------------------------
INT 62 - PC-DRAFT - PRIMARY DISPLAY DRIVER
  ???
Return: ???
Program: PC-DRAFT is a powerful CAD environment by rhv.
SeeAlso: INT 64"PC-DRAFT",INT 65"PC-DRAFT",INT 66"PC-DRAFT",INT 67"PC-DRAFT"
--------N-6200------------------------------
INT 62 - BW-TCP - HW DRIVER (ETHDEV.SYS) - GET PHYSICAL HARDWARE ADDRESS
  AH = 00h
  DS:DX -> 6-byte buffer for address
Return: AX = length of hardware address???
Range:  INT 4C to INT FB, selected by configuration
Note: the Beame&Whiteside TCP/IP protocol stack uses two consecutive
    interrupts (62h and 63h by default); the BW-NFS client uses a third
    consecutive interrupt (64h by default) if it is loaded
SeeAlso: AH=01h"ETHDEV",AH=04h"ETHDEV",AH=09h"ETHDEV",AH=0Eh"ETHDEV"
SeeAlso: AH=14h"ETHDEV",AH=18h"ETHDEV",AH=FEh,INT 21/AH=3Fh"BW-TCP"
SeeAlso: INT 63/AH=03h"BW-TCP",INT 63/AH=24h"BW-TCP",INT 64/AH=01h"BW-NFS"
--------V-620000------------------------------
INT 62 u - FGDRIVER v3.03+ - "FG_ALLOCATE" - CREATE VIRTUAL VIDEO PAGE
  AX = 0000h
  BX = page number (0000h-003Fh)
Return: AX = status (0000h,0001h,0007h,0008h) (see #03473)
Program: FGDRIVER is the external video driver for the shareware
    Fastgraph/Light by Ted Gruber Software
InstallCheck: test for the signature "FG" ten bytes beyond the start of the
    interrupt handler
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
  The amount of memory required by the virtual video page depends on the
    current video mode
SeeAlso: AX=0001h,AX=0002h,AX=0003h,AX=0038h
Index:  installation check;Fastgraph/Light
```

```
(Table 03473)
Values for FGDRIVER status:
 0000h  successful
 0001h  specified page is a physical or logical page
 0007h  virtual page created/released, but memory control blocks corrupted
 0008h  not enough memory
 0009h  attempt to free a page which was never created
 FFFCh  insufficient memory
 FFFDh  page already created, or exists as physical or virtual page
 FFFEh  invalid page number
 FFFFh  memory manager not initialized
SeeAlso: #03474

(Table 03474)
Values for FGDRIVER functions (by driver version):
Name    v1.10 v3.03 v4.02
 FG_ALLOCATE  0042h 0000h 0000h
 FG_ALLOCCMS  009Dh 0001h 0001h
 FG_ALLOCEMS  009Eh 0002h 0002h
 FG_ALLOCXMS  009Fh 0003h 0003h
 FG_AUTOMODE  0004h 0004h 0004h
 FG_BESTMODE  0003h 0005h 0005h
 FG_BOX   00A2h 0006h 0006h
 FG_BOXDEPTH  00A3h 0007h 0007h
 FG_BOXX   -- 0008h 0008h
 FG_BUTTON  0078h 0009h 0009h
 FG_CAPSLOCK  0070h 000Ah 000Ah
 FG_CHGATTR 0035h 000Bh 000Bh
 FG_CHGTEXT 0036h 000Ch 000Ch
 FG_CIRCLE  0030h 000Dh 000Dh
 FG_CIRCLEF  -- 000Eh 000Eh
 FG_CLIPMAP  --  -- 000Fh
 FG_CLIPMASK  0052h 000Fh 0010h
 FG_CLPIMAGE  004Ah 0010h 0011h
 FG_CLPRECT 002Bh 0011h 0012h
 FG_COLORS   --  -- 0013h
 FG_COPYPAGE  005Fh 0012h 0014h
 FG_CURSOR  0007h 0013h 0015h
 FG_DASH  0027h 0014h 0016h
 FG_DASHREL 0028h 0015h 0017h
```

```
FG_DEFCOLOR  0014h 0016h 0018h
FG_DEFPAGES   -- 0017h 0019h
FG_DISPFILE  004Fh 0018h 001Ah
FG_DISPLAY 004Dh 0019h 001Bh
FG_DISPLAYP  004Eh 001Ah 001Ch
FG_DISPPCX 0060h  --  --
FG_DRAW  0025h 001Bh 001Dh
FG_DRAWMAP 0047h 001Ch 001Eh
FG_DRAWMASK  0051h 001Dh 001Fh
FG_DRAWREL 0026h 001Eh 0020h
FG_DRAWRELX   -- 001Fh 0021h
FG_DRAWX  -- 0020h 0022h
FG_DRECT 002Ch 0021h 0023h
FG_DRWIMAGE  0049h 0022h 0024h
FG_EGACHECK  0005h 0023h 0025h
FG_ELLIPSE 002Dh 0024h 0026h
FG_ELLIPSEF   -- 0025h 0027h
FG_ERASE 001Eh 0026h 0028h
FG_FADEIN  0064h 0027h 0029h
FG_FADEOUT 0065h 0028h 002Ah
FG_FILLPAGE   -- 0029h 002Bh
FG_FINDPAGE   --  -- 002Ch
FG_FLICDONE   --  -- 002Dh
FG_FLICHEAD   --  -- 002Eh
FG_FLICMODE   --  -- 002Fh
FG_FLICOPEN   --  -- 0030h
FG_FLICPLAY   --  -- 0031h
FG_FLICSIZE   --  -- 0032h
FG_FLICSKIP   --  -- 0033h
FG_FLIPMASK  0054h 002Ah 0034h
FG_FLOOD  -- 002Bh 0035h
FG_FLPIMAGE  004Ch 002Ch 0036h
FG_FONTSIZE   -- 002Dh 0037h
FG_FREEPAGE  0043h 002Eh 0038h
FG_GETADDR 0044h 002Fh 0039h
FG_GETATTR 0037h 0030h 003Ah
FG_GETBANKS   --  -- 003Bh
FG_GETBLOCK   -- 0031h 003Ch
FG_GETCHAR 0038h 0032h 003Dh
FG_GETCLIP  --  -- 003Eh
FG_GETCLOCK  0099h 0033h 003Fh
```

```
FG_GETCOLOR  0015h 0034h 0040h
FG_GETDACS 00A4h 0035h 0041h
FG_GETENTRY   -- 0036h 0042h
FG_GETHPAGE  0040h 0037h 0043h
FG_GETIMAGE  0048h 0038h 0044h
FG_GETINDEX  0016h 0039h 0045h
FG_GETKEY  006Eh 003Ah 0046h
FG_GETLINES  0010h 003Bh 0047h
FG_GETMAP  0046h 003Ch 0048h
FG_GETMAXX 000Ah 003Dh 0049h
FG_GETMAXY 000Bh 003Eh 004Ah
FG_GETMODE 0000h 003Fh 004Bh
FG_GETPAGE 003Ch 0040h 004Ch
FG_GETPIXEL  0020h 0041h 004Dh
FG_GETRGB  001Ch 0042h 004Eh
FG_GETVIEW  --  -- 004Fh
FG_GETVPAGE  003Eh 0043h 0050h
FG_GETXBOX  --  -- 0051h
FG_GETXJOY 0076h 0044h 0052h
FG_GETXJUST   --  -- 0053h
FG_GETXPOS 0021h 0045h 0054h
FG_GETYBOX  --  -- 0055h
FG_GETYJOY 0077h 0046h 0056h
FG_GETYJUST   --  -- 0057h
FG_GETYPOS 0022h 0047h 0058h
FG_HUSH  008Eh 0048h 0059h
FG_HUSHNEXT  008Fh 0049h 005Ah
FG_IMAGEBUF   -- 004Ah 005Bh
FG_IMAGESIZ  0062h 004Bh 005Ch
FG_INITEMS 00A0h 004Ch 005Dh
FG_INITJOY 0075h 004Dh 005Eh
FG_INITPM   --  -- 005Fh??? (NOP in v4.02)
FG_INITXMS 00A1h 004Eh 0060h
FG_INSIDE   -- 004Fh 0061h
FG_INTJOY  0079h 0050h 0062h
FG_INTKEY  006Fh 0051h 0063h
FG_INVERT   --  -- 0064h
FG_JUSTIFY  -- 0052h 0065h
FG_KBINIT   -- 0053h 0066h
FG_KBLAST   --  -- 0067h
FG_KBRESET  --  -- 0068h
```

```
FG_KBTEST   -- 0054h 0069h
FG_LOADPCX  --  -- 006Ah
FG_LOCATE  0033h 0055h 006Bh
FG_MAKEPCX 0061h 0057h 006Ch
FG_MAKEPPR  -- 0058h 006Dh
FG_MAKESPR  -- 0059h 006Eh
FG_MAPRGB  001Dh 005Ah 006Fh
FG_MEASURE 0098h 005Bh 0070h
FG_MEMAVAIL  009Bh 005Ch 0071h
FG_MEMORY   -- 005Dh 0072h
FG_MOUSE256   --  -- 0073h
FG_MOUSEBUT  007Ch 005Eh 0074h
FG_MOUSECUR  007Eh 005Fh 0075h
FG_MOUSEFIN   -- 0060h 0076h
FG_MOUSEINI  007Ah 0061h 0077h
FG_MOUSELIM  0080h 0062h 0078h
FG_MOUSEMOV  0081h 0063h 0079h
FG_MOUSEPOS  007Dh 0064h 007Ah
FG_MOUSEPTR  007Fh 0065h 007Bh
FG_MOUSESPD  0082h 0066h 007Ch
FG_MOUSEVIS  007Bh 0067h 007Dh
FG_MOVE  0023h 0068h 007Eh
FG_MOVEREL 0024h 0069h 007Fh
FG_MUSIC 008Ch 006Ah 0080h
FG_MUSICB  008Dh 006Bh 0081h
FG_NUMLOCK 0072h 006Ch 0082h
FG_PACK   --  -- 0083h
FG_PAGESIZE   -- 006Dh 0084h
FG_PAINT 002Fh 006Eh 0085h
FG_PALETTE 0017h 006Fh 0086h
FG_PALETTES  0018h 0070h 0087h
FG_PAN   0066h 0071h 0088h
FG_PATTERN 0050h 0072h 0089h
FG_PCXHEAD  -- 0073h 008Ah
FG_PCXMODE  -- 0074h 008Bh
FG_PCXPAL   --  -- 008Ch
FG_PCXRANGE   --  -- 008Dh
FG_PLAYING 0091h 0075h 008Eh
FG_POINT 001Fh 0076h 008Fh
FG_POINTX   -- 0077h 0090h
FG_POLYEDGE   --  -- 0091h
```

```
FG_POLYFILL   -- 0078h 0092h
FG_POLYGON 002Eh 0079h 0093h
FG_POLYLINE   -- 007Ah 0094h
FG_POLYOFF  -- 007Bh 0095h
FG_PRINT  -- 007Ch 0096h
FG_PRINTC   --  -- 0097h
FG_PUTBLOCK   -- 007Dh 0098h
FG_PUTIMAGE   -- 007Eh 0099h
FG_QUIET 0090h 007Fh 009Ah
FG_RECT  002Ah 0080h 009Bh
FG_RESET 0006h 0081h 009Ch
FG_RESIZE  0045h 0082h 009Dh
FG_RESTORE 005Bh 0083h 009Eh
FG_RESUME  0092h 0084h 009Fh
FG_REVIMAGE  004Bh 0085h 00A0h
FG_REVMASK 0053h 0086h 00A1h
FG_SAVE   -- 0087h 00A2h
FG_SCALE  --  -- 00A3h
FG_SCRLOCK 0074h 0088h 00A4h
FG_SCROLL  0067h 0089h 00A5h
FG_SETATTR 0019h 008Ah 00A6h
FG_SETBANKS   --  -- 00A7h
FG_SETCAPS 0071h 008Bh 00A8h
FG_SETCLIP 0029h 008Ch 00A9h
FG_SETCOLOR  001Ah 008Dh 00AAh
FG_SETDACS 00A5h 008Eh 00ABh
FG_SETENTRY   -- 008Fh 00ACh
FG_SETFUNC 009Ch 0090h 00ADh
FG_SETHPAGE  0041h 0091h 00AEh
FG_SETLINES  0011h 0092h 00AFh
FG_SETMODE 0001h 0093h 00B0h
FG_SETNUM  0073h 0094h 00B1h
FG_SETPAGE 003Dh 0095h 00B2h
FG_SETRGB  001Bh 0096h 00B3h
FG_SETVIEW  --  -- 00B5h
FG_SETVPAGE  003Fh 0098h 00B6h
FG_SHEAR  --  -- 00B7h
FG_SHOWFLIC   --  -- 00B8h
FG_SHOWPCX  -- 009Ah 00B9h
FG_SHOWPPR  -- 009Bh 00BAh
FG_SHOWSPR  -- 009Ch 00BBh
```

```
FG_SOUND 0088h 009Dh 00BCh
FG_SOUNDS  0089h 009Eh 00BDh
FG_SPLIT  --  -- 00BEh
FG_STALL 0097h 009Fh 00BFh
FG_SUSPEND 0093h 00A0h 00C0h
FG_SVGAINIT   -- 00A1h 00C1h
FG_SVGASTAT   -- 00A2h 00C2h
FG_SVGAVER  -- 00A3h 00C3h
FG_TCDEFINE   -- 00A4h 00C4h
FG_TCMASK  005Eh 00A5h 00C5h
FG_TCXFER  005Dh 00A6h 00C6h
FG_TESTMODE  0002h 00A7h 00C7h
FG_TEXT  0032h 00A8h 00C8h
FG_TEXTC  --  -- 00C9h
FG_TRANSFER  005Ch 00A9h 00CAh
FG_UNPACK   --  -- 00CBh
FG_VBADDR   --  -- 00CCh
FG_VBALLOC  --  -- 00CDh
FG_VBCLOSE  --  -- 00CEh
FG_VBCOPY   --  -- 00CFh
FG_VBCUT  --  -- 00D0h
FG_VBDEFINE   --  -- 00D1h
FG_VBFREE   --  -- 00D2h
FG_VBHANDLE   --  -- 00D3h
FG_VBINIT   --  -- 00D4h
FG_VBOPEN   --  -- 00D5h
FG_VBPASTE  --  -- 00D6h
FG_VBTCCOPY   --  -- 00D7h
FG_VBTCXFER   --  -- 00D8h
FG_VBUNDEF  --  -- 00D9h
FG_VGASTATE   --  -- 00DAh
FG_VOICE 008Ah 00AAh 00DBh
FG_VOICES  008Bh 00ABh 00DCh
FG_WAITFOR 0096h 00ACh 00DDh
FG_WAITKEY 006Dh 00ADh 00DEh
FG_WAITVR   -- 00AEh 00DFh
FG_WHERE 0034h 00AFh 00E0h
FG_XALPHA  000Ch 00B0h 00E1h
FG_XCONVERT  000Eh 00B1h 00E2h
FG_XVIEW  --  -- 00E3h
FG_YALPHA  000Dh 00B2h 00E4h
```

```
 FG_YCONVERT  000Fh 00B3h 00E5h
 FG_YVIEW  --  -- 00E6h
SeeAlso: #03473
--------V-620001-----------------------------
INT 62 u - FGDRIVER v3.03+ - "FG_ALLOCCMS" - CREATE LOGICAL VIDEO PAGE (CONV)
  AX = 0001h
  BX = page number (0001h-003Fh)
Return: AX = status (0000h,FFFCh,FFFDh,FFFEh) (see #03473)
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
  The only operation which is allowed on logical pages is "FG_COPYPAGE"
    (see AX=0014h)
SeeAlso: AX=0000h,AX=0002h,AX=0003h,AX=0014h,AX=0038h
--------V-620002-----------------------------
INT 62 u - FGDRIVER v3.03+ - "FG_ALLOCEMS" - CREATE LOGICAL VIDEO PAGE (EMS)
  AX = 0002h
  BX = page number (0001h-003Fh)
Return: AX = status (0000h,FFFCh,FFFDh,FFFEh) (see #03473)
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
  You must first call "FG_INITEMS" (see AX=005Dh) before using this
    function
  The only operation which is allowed on logical pages is "FG_COPYPAGE"
    (see AX=0014h)
SeeAlso: AX=0000h,AX=0001h,AX=0003h,AX=0014h,AX=0038h,AX=005Dh
--------V-620003-----------------------------
INT 62 u - FGDRIVER v3.03+ - "FG_ALLOCXMS" - CREATE LOGICAL VIDEO PAGE (XMS)
  AX = 0003h
  BX = page number (0001h-003Fh)
Return: AX = status (0000h,FFFCh,FFFDh,FFFEh) (see #03473)
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
  You must first call "FG_INITXMS" (see AX=0060h) before using this
    function
  The only operation which is allowed on logical pages is "FG_COPYPAGE"
    (see AX=0014h)
SeeAlso: AX=0000h,AX=0001h,AX=0002h,AX=0014h,AX=0038h,AX=0060h
--------V-620004-----------------------------
INT 62 u - FGDRIVER 1.10,3.03+ - "FG_AUTOMODE" - GET VIDEO MODE W/MOST FEATURES
  AX = 0004h
Return: AX = proposed video mode number (see #03500 at AX=00B0h)
```

```
Program: FGDRIVER is the external video driver for the shareware
    Fastgraph/Light by Ted Gruber Software
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
SeeAlso: AX=0005h,AX=004Bh,AX=00B0h,AX=00C1h,AX=00C7h
--------V-620005-----------------------------
INT 62 u - FGDRIVER v3.03+ - "FG_BESTMODE" - GET BEST VIDEO MODE GIVEN RESOLUTN
  AX = 0005h
  BX = horizontal resolution
  CX = vertical resolution
  DX = number of video pages required (both physical and virtual)
Return: AX = proposed video mode number or FFFFh if no matching video mode
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
  You must first call "FG_SVGAINIT" (see AX=00C1h) to get SVGA video
    modes
SeeAlso: AX=0004h,AX=004Bh,AX=00B0h,AX=00C1h,AX=00C7h
--------V-620006-----------------------------
INT 62 u - FGDRIVER v3.03+ - "FG_BOX" - DRAW UNFILLED RECTANGLE
  AX = 0006h
  BX = left column
  CX = right column
  DX = top row
  SI = bottom row
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
  The rectangle is drawn in screen space, respecting the clipping region,
    with edges of the width specified with "FG_BOXDEPTH" (see AX=0007h)
    (default = 1 is set by "FG_SETMODE")
  This function has no effect in text modes
SeeAlso: AX=0007h,AX=0008h,AX=000Ch,AX=0016h,AX=001Dh,AX=0026h,AX=0051h
SeeAlso: AX=0055h,AX=008Fh
--------V-620007-----------------------------
INT 62 u - FGDRIVER v3.03+ - "FG_BOXDEPTH" - SET RECTANGLE BORDER WIDTH
  AX = 0007h
  BX = width of left and right edges in pixels (> 0)
  CX = width of top and bottom edges in pixels (> 0)
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
SeeAlso: AX=0006h,AX=0008h,AX=0051h,AX=0055h
--------V-620008-----------------------------
```

```
INT 62 u - FGDRIVER v3.03+ - "FG_BOXX" - XOR HOLLOW RECTANGLE
  AX = 0008h
  BX = left column
  CX = right column
  DX = top row
  SI = bottom row
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
  The rectangle is XORed into screen space, respecting the clipping
    region, with edges of the width specified with "FG_BOXDEPTH"
    (see AX=0007h) (default = 1 is set by "FG_SETMODE")
  This function has no effect in text modes
SeeAlso: AX=0006h,AX=0007h,AX=0021h,AX=0022h,AX=0051h,AX=0055h,AX=0090h
--------V-620009-----------------------------
INT 62 u - FGDRIVER v3.03+ - "FG_BUTTON" - GET JOYSTICK BUTTON STATE
  AX = 0009h
  BX = joystick number (0001h or 0002h)
Return: AX = button states
      bit 0: top button pressed
      bit 1: bottom button pressed
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
SeeAlso: AX=0052h,AX=0056h,AX=005Eh,AX=0062h,AX=0077h
--------V-62000A-----------------------------
INT 62 u - FGDRIVER v3.03+ - "FG_CAPSLOCK" - GET STATE OF CAPSLOCK KEY
  AX = 000Ah
Return: AX = CapsLock state (0000h off, 0001h on)
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
SeeAlso: AX=0082h,AX=00A4h,AX=00A8h,AX=00B1h
--------V-62000B-----------------------------
INT 62 u - FGDRIVER v3.03+ - "FG_CHGATTR" - APPLY CURRENT TEXT ATTRIB TO CHARS
  AX = 000Bh
  BX = number of characters to recolor
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
  This call is ignored in graphics modes
  Starting at the current text cursor position, the specified number of
    characters have their attributes to the current text attribute
SeeAlso: AX=000Ch,AX=003Ah,AX=00A6h
--------V-62000C-----------------------------
```

```
INT 62 u - FGDRIVER v3.03+ - "FG_CHGTEXT" - DISPLAY STRING AT CURSOR POSITION
  AX = 000Ch
  CX = length of string
  ES:BX -> string to be displayed
Return: text cursor updated
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
  This call is ignored in graphics modes
SeeAlso: AX=000Bh,AX=0096h
--------V-62000D-----------------------------
INT 62 u - FGDRIVER v3.03+ - "FG_CIRCLE" - DRAW UNFILLED CIRCLE
  AX = 000Dh
  BX = radius in horizontal screen space units (> 0)
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
  The circle is drawn in screen space, centered at the current graphics
    cursor position
  This call is ignored in text modes
SeeAlso: AX=0006h,AX=0008h,AX=000Eh,AX=0026h,AX=0093h
--------V-62000E-----------------------------
INT 62 u - FGDRIVER v3.03+ - "FG_CIRCLEF" - DRAW FILLED CIRCLE
  AX = 000Eh
  BX = radius in horizontal screen space units (> 0)
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
SeeAlso: AX=0008h,AX=000Dh,AX=0027h,AX=0092h
--------V-62000F-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_CLIPMAP" - DISPLAY CLIPPED IMAGE (MODE-INDEP)
  AX = 000Fh
  ES:BX -> bitmap
  CX = width
  DX = height
Desc: display a mode-independent bitmap, showing only the portion within
    the current clipping limits
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
  This call is ignored in text modes and in native EGA and VGA graphics
    modes
  The image is drawn with its lower left corner at the current graphics
    cursor position
SeeAlso: AX=0011h,AX=001Fh,AX=0034h,AX=0086h,AX=00A9h,AX=00C5h
```

```
--------V-620010-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_CLIPMASK" - DISPLAY CLIPPED IMAGE (MASKING MAP)
  AX = 0010h
  ES:BX -> array containing image stored as a masking map (see #03475)
  CX = number of pixel runs in masking map
  DX = width of masking map in pixels
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
  This call is ignored in text modes and in native EGA and VGA graphics
    modes
  The image is drawn with its lower left corner at the current graphics
    cursor position
SeeAlso: AX=0011h,AX=001Fh,AX=0034h,AX=00A1h,AX=00A9h,AX=00C5h
--------V-620011-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_CLPIMAGE" - DISPLAY CLIPPED IMAGE (BITMAP)
  AX = 0011h
  ES:BX -> mode-specific bitmap
  CX = width of bit map in bytes
  DX = height of bit map in pixel rows
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
  This call is ignored in text modes
  The image is drawn with its lower left corner at the current graphics
    cursor position
  The current clipping region is used, extended to a byte boundary
SeeAlso: AX=0010h,AX=0024h,AX=0036h,AX=0044h,AX=00A0h,AX=00A9h,AX=00B9h
--------V-620012-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_CLPRECT" - DRAW FILLED RECTANGLE IN SCREEN SPCE
  AX = 0012h
  BX = screen space column of left edge
  CX = screen space column of right edge
  DX = screen space row of top edge
  SI = screen space row of bottom edge
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
  This call is ignored in text modes
SeeAlso: AX=0006h,AX=0010h,AX=0011h,AX=0023h,AX=009Bh,AX=00A9h
--------V-620013-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_COLORS" - GET SIMULTANEOUSLY-AVAILABLE COLORS
  AX = 0013h
Return: AX = number of colors available at one time
```

```
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
--------V-620014-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_COPYPAGE" - TRANSFER VIDEO PAGE CONTENTS
  AX = 0014h
  BX = source page number (0000h-003Fh)
  CX = destination page number (0000h-003Fh)
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
  If both source and destination pages are logical pages, they must both
    be located in the same type (conventional, EMS, XMS) of memory
SeeAlso: AX=009Eh,AX=00A2h,AX=00C6h,AX=00CAh
--------V-620015-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_CURSOR" - SPECIFY WHETHER TEXT CURSR IS VISIBLE
  AX = 0015h
  BX = new state (0000h invisible, 0001h visible)
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
  This call is ignored in text modes
SeeAlso: AX=000Ch,AX=0054h,AX=0058h,AX=006Bh,AX=0075h,AX=0079h,AX=007Eh
SeeAlso: AX=00E0h
--------V-620016-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_DASH" - DRAW DASHED LINE TO ABSOLUTE POSITION
  AX = 0016h
  BX = endpoint column
  CX = endpoint row
  DX = dash pattern (set bits cause drawn pixels)
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
  This call is ignored in text modes
  The starting point is the current graphics cursor position; the cursor
    position is updated
SeeAlso: AX=0017h,AX=001Dh,AX=008Fh
--------V-620017-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_DASHREL" - DRAW DASHED LINE TO RELATVE POSITION
  AX = 0017h
  BX = endpoint column offset
  CX = endpoint row offset
  DX = dash pattern (set bits cause drawn pixels)
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
```

```
  This call is ignored in text modes
  The starting point is the current graphics cursor position; the cursor
    position is updated
SeeAlso: AX=0016h,AX=001Eh,AX=008Fh
--------V-620018-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_DEFCOLOR" - ASSIGN COLOR VALUE TO COLOR INDEX
  AX = 0018h
  BX = color index (0000h-00FFh)
  CX = new color value (0 to maximum color value for current video mode)
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
  This call is ignored in text modes and 256-color graphics modes
SeeAlso: AX=0040h,AX=0045h,AX=00AAh
--------V-620019-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_DEFPAGES" - DEF SRC/DEST PAGES FOR BLOCK XFERS
  AX = 0019h
  BX = source page
  CX = destination page
Desc: specify the source and destination SVGA banks for block transfers on
    extended video pages
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
  This function is ignored if the video controller does not support
    extended pages or the current video mode does not allow them
SeeAlso: AX=0014h,AX=00CAh
--------V-62001A-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_DISPFILE" - DISPLAY STORED IMAGE
  AX = 001Ah
  ES:BX -> ASCIZ filename
  CX = image width in pixels (> 0)
  DX = image format
       0000h Fastgraph standard pixel run format
       0001h packed pixel run format
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
  This call is ignored in text modes
  The image is displayed with its lower left corner at the current
    graphics cursor position
SeeAlso: AX=001Bh,AX=001Ch,AX=00B9h,AX=00BAh,AX=00BBh
--------V-62001B-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_DISPLAY" - DISPLAY IMAGE (STD PIXEL RUN FORMAT)
```

```
  AX = 001Bh
  ES:BX -> pixel run map (pairs of bytes: color index, count)
  CX = number of pixel runs to display
  DX = width of image in pixels (> 0)
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
  This call is ignored in text modes
  The image is displayed with its lower left corner at the current
    graphics cursor position
SeeAlso: AX=001Ah,AX=001Ch,AX=00B9h,AX=00BAh,AX=00BBh
--------V-62001C-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_DISPLAYP" - DISPLAY IMAGE (PACKED PIXEL RUNS)
  AX = 001Ch
  ES:BX -> pixel run map (trios of bytes: colors, count1, count2; colors
      contains the color for the first run in its high nybble and
      the color for the second run in its low nybble)
  CX = number of pixel runs to display
  DX = width of image in pixels (> 0)
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
  This call is ignored in text modes
  The image is displayed with its lower left corner at the current
    graphics cursor position
SeeAlso: AX=001Ah,AX=001Bh,AX=00B9h,AX=00BAh,AX=00BBh
--------V-62001D-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_DRAW" - DRAW SOLID LINE TO ABSOLUTE POSITION
  AX = 001Dh
  BX = endpoint column
  CX = endpoint row
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
  This call is ignored in text modes
  The starting point is the current graphics cursor position; the cursor
    position is updated
SeeAlso: AX=0016h,AX=0020h,AX=0021h,AX=0026h,AX=008Fh,AX=0090h
--------V-62001E-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_DRAWMAP" - DISPLAY MODE-INDEPENDENT BIT MAP
  AX = 001Eh
  ES:BX -> bitmap (each set bit is pixel drawn in current color)
  CX = width of bitmap in bytes
  DX = height of bitmap in pixel rows
```

```
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
SeeAlso: AX=0011h,AX=0024h,AX=0036h,AX=0044h,AX=0099h,AX=00A0h
--------V-62001F-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_DRAWMASK" - DISPLAY IMAGE (MASKING MAP)
  AX = 001Fh
  ES:BX -> array containing image stored as a masking map (see #03475)
  CX = number of pixel runs in masking map
  DX = width of masking map in pixels
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
  This call is ignored in text modes and in native EGA and VGA graphics
    modes
  The image is drawn with its lower left corner at the current graphics
    cursor position
SeeAlso: AX=0010h,AX=001Fh,AX=00A1h,AX=00A9h

Format of FGDRIVER masking map:
Offset  Size  Description (Table 03475)
 00h  BYTE  length of first "protect" run (pixels remain unchanged)
 01h  BYTE  length of first "zero" run (pixels set to background color)
 02h  BYTE  length of second "protect" run
 03h  BYTE  length of second "zero" run
  ...
--------V-620020-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_DRAWREL" - DRAW SOLID LINE TO RELATIVE POSITION
  AX = 0020h
  BX = endpoint column offset
  CX = endpoint row offset
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This call is ignored in text modes
  The starting point is the current graphics cursor position; the cursor
    position is updated
SeeAlso: AX=0006h,AX=000Dh,AX=001Dh,AX=0021h,AX=0026h,AX=008Fh
--------V-620021-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_DRAWRELX" - XOR SOLID LINE TO RELATIVE POSITION
  AX = 0021h
  BX = endpoint column offset
  CX = endpoint row offset
Desc: draw a solid line, XORing each pixel onto the screen
```

```
Notes:  the FGDRIVER functions are rearranged with each major release, but
     their parameters do not change (see #03474 at AX=0000h)
  This call is ignored in text modes
  The starting point is the current graphics cursor position; the cursor
     position is updated
SeeAlso: AX=0006h,AX=000Dh,AX=001Dh,AX=0020h,AX=0026h,AX=0090h
--------V-620022-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_DRAWX" - XOR SOLID LINE TO ABSOLUTE POSITION
  AX = 0022h
  BX = endpoint column
  CX = endpoint row
Desc: draw a solid line, XORing each pixel onto the screen
Notes:  the FGDRIVER functions are rearranged with each major release, but
     their parameters do not change (see #03474 at AX=0000h)
  This call is ignored in text modes
  The starting point is the current graphics cursor position; the cursor
     position is updated
SeeAlso: AX=001Dh,AX=0021h,AX=008Fh
--------V-620023-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_DRECT" - DRAW DITHERED RECTANGLE IN SCRN SPACE
  AX = 0023h
  BX = screen space column of left edge
  CX = screen space column of right edge
  DX = screen space row of top edge
  SI = screen space row of bottom edge
  ES:DI -> dithering matrix (video-mode dependent)
Notes:  the FGDRIVER functions are rearranged with each major release, but
     their parameters do not change (see #03474 at AX=0000h)
  This call is ignored in text modes
SeeAlso: AX=0006h,AX=0008h,AX=0013h,AX=003Ch,AX=0089h,AX=009Bh
--------V-620024-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_DRWIMAGE" - DISPLAY BITMAPPED IMAGE
  AX = 0024h
  ES:BX -> video mode-specific bitmap
  CX = width of bitmap in bytes
  DX = height of bitmap in pixel rows
Notes:  the FGDRIVER functions are rearranged with each major release, but
     their parameters do not change (see #03474 at AX=0000h)
  The image will be drawn with its lower left corner at the current
     cursor position (either text or graphics)
SeeAlso: AX=0011h,AX=0036h,AX=0044h,AX=0099h,AX=00A0h
```

```
--------V-620025-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_EGACHECK" - GET INFO ABOUT ACTIVE EGA DISPLAY
  AX = 0025h
Return: AX = number of 64K banks of video memory, or 0000h if no EGA or EGA
        without an Enhanced Color Display
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=00C2h
--------V-620026-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_ELLIPSE" - DRAW UNFILLED ELLIPSE IN SCRN SPACE
  AX = 0026h
  BX = horizontal semi-axis length in screen space units
  CX = vertical semi-axis length in screen space units
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This call is ignored in text modes
  The ellipse is centered at the current graphics cursor position
SeeAlso: AX=0006h,AX=000Dh,AX=0027h,AX=008Fh,AX=0093h
--------V-620027-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_ELLIPSEF" - DRAW FILLED ELLIPSE IN SCREEN SPACE
  AX = 0027h
  BX = horizontal semi-axis length in screen space units
  CX = vertical semi-axis length in screen space units
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This call is ignored in text modes
  The ellipse is centered at the current graphics cursor position
SeeAlso: AX=0006h,AX=000Eh,AX=0026h,AX=008Fh,AX=0092h
--------V-620028-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_ERASE" - CLEAR THE ACTIVE VIDEO PAGE
  AX = 0028h
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This function sets each pixel to 0 in graphics modes, each character
    cell to a blank with a gray foreground attribute in text modes
SeeAlso: AX=0029h,AX=002Bh,AX=0035h
--------V-620029-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_FADEIN" - FADE IN HIDDEN PAGE
  AX = 0029h
  BX = delay (0000h = fastest possible fade-in)
Notes:  the FGDRIVER functions are rearranged with each major release, but
```

```
    their parameters do not change (see #03474 at AX=0000h)
  The current hidden page is copied to the current visible page in small
    random sections to produce a fade-in effect
  This call is ignored in text modes
SeeAlso: AX=0028h,AX=002Ah,AX=002Bh
--------V-62002A-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_FADEOUT" - FADE OUT TO CURRENT COLOR
  AX = 002Ah
  BX = delay (0000h = fastest possible fade-out)
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  The current visible page is filled with pixels of the current color in
    small random sections to give a fade-out effect
  This call is ignored in text modes
SeeAlso: AX=0028h,AX=0029h,AX=002Bh
--------V-62002B-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_FILLPAGE" - FILL THE ACTIVE VIDEO PAGE
  AX = 002Bh
Desc: fill the active video page with pixels of the current color (graphics
    modes) or the block character DBh with the current character
    attributes (text modes)
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0028h,AX=0029h,AX=002Ah
--------V-62002C-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_FINDPAGE" - FIND AVAILABLE VIRTUAL/LOGICAL PAGE
  AX = 002Ch
Return: AX = first available page number (virtual or logical page)
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0028h
--------V-62002D-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_FLICDONE" - CLOSE FLIC FILE
  AX = 002Dh
  ES:BX -> context descriptor (see AX=0030h)
Return: nothing
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=002Eh,AX=0030h
--------V-62002E-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_FLICHEAD" - READ FLI/FLC FILE HEADER
```

```
  AX = 002Eh
  ES:BX -> FLICHEAD variable pointer record (see #03477)
Return: AX = status (see #03476)
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=002Dh,AX=002Fh,AX=0030h,AX=0032h

(Table 03476)
Values for FGDRIVER FLIC processing status:
 0000h  successful
 FFFEh  not an FLI or FLC file
 FFFFh  file not found

Format of FGDRIVER FLICHEAD variable pointer record:
Offset  Size  Description (Table 03477)
 00h  WORD  segment of ASCIZ FLI/FLC file name
 02h  WORD  offset of ASCIZ FLI/FLC file name
 04h  WORD  segment of buffer for 128-byte file header
 06h  WORD  offset of buffer for 128-byte file header
--------V-62002F-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_FLICMODE" - GET OPTIMAL VIDEO MODE FOR FLI/FLC
  AX = 002Fh
  ES:BX -> 128-byte buffer containing FLI/FLC file header (see AX=002Eh)
Return: AX = optimal 256-color graphics mode number
      FFFFh if invalid file header
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=002Eh,AX=0030h
--------V-620030-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_FLICOPEN" - OPEN FLI/FLC FILE
  AX = 0030h
  ES:BX -> FLICOPEN variable pointer record (see #03478)
Return: AX = status (see #03476)
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=002Dh,AX=002Eh,AX=002Fh,AX=0030h,AX=0031h,AX=0032h

Format of FGDRIVER FLICOPEN variable pointer record:
Offset  Size  Description (Table 03478)
 00h  WORD  segment of ASCIZ FLI/FLC file name
 02h  WORD  offset of ASCIZ FLI/FLC file name
```

```
 04h  WORD  segment of buffer for 16-byte file context descriptor
 06h  WORD  offset of buffer for 16-byte file context descriptor
--------V-620031-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_FLICPLAY" - DISPLAY NEXT FRAME(S) IN FLI/FLC
  AX = 0031h
  ES:BX -> context descriptor (see AX=0030h)
  CX = number of frames to display starting at current frame
  DX = control flags (see #03479)
Return: AX = number of frames displayed
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0030h,AX=0033h,AX=00B8h

Bitfields for FGDRIVER FLI/FLC control flags:
Bit(s)  Description (Table 03479)
 0  skip inter-frame delay specified in FLI/FLC header
 1  display relative to current graphics position
 2  dipslay image from the FG_IMAGEBUF buffer instead of file
--------V-620032-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_FLICSIZE" - GET IMAGE SIZE
  AX = 0032h
  ES:BX -> FLICSIZE variable pointer record (see #03480)
Return: image height/width buffers updated; height is set to FFFFh on error
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=002Eh,AX=0030h

Format of FGDRIVER FLICSIZE variable pointer record:
Offset  Size  Description (Table 03480)
 00h  WORD  segment of buffer for 128-byte FLI/FLC file header
 02h  WORD  offset of buffer for 128-byte FLI/FLC file header
 04h  WORD  segment of WORD buffer for image width in pixels
 06h  WORD  offset of WORD buffer for image width in pixels
 08h  WORD  segment of WORD buffer for image height in pixels
 0Ah  WORD  offset of WORD buffer for image height in pixels
--------V-620033-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_FLICSKIP" - SKIP FRAME(S) IN FLI/FLC FILE
  AX = 0033h
  ES:BX -> context descriptor (see AX=0030h)
  CX = number of frames to skip
       reset to first frame if skip count is negative
```

```
Return: AX = number of frames skipped (may be less than requested if EOF)
      0000h if resetting to first frame
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0030h,AX=0031h
--------V-620034----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_FLIPMASK" - DISPLAY INV CLIPPED IMAGE (MASKMAP)
  AX = 0034h
  ES:BX -> array containing image stored as a masking map (see #03475)
  CX = number of pixel runs in masking map
  DX = width of masking map in pixels
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This call is ignored in text modes and in native EGA and VGA graphics
    modes
  The image is drawn with its lower left corner at the current graphics
    cursor position
SeeAlso: AX=000Fh,AX=001Fh,AX=0036h,AX=00A1h,AX=00A5h,AX=00C5h
--------V-620035----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_FLOOD" - FLOOD FILL ARBITRARY CLOSED REGION
  AX = 0035h
  BX = starting column
  CX = starting row
Desc: fill the bounded region around the specified point (respecting clipping
    region) with the currrent color
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This function is ignored in text modes
SeeAlso: AX=0028h,AX=0085h
--------V-620036----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_FLPIMAGE" - DISPLAY INV CLIPPED IMAGE (BITMAP)
  AX = 0036h
  ES:BX -> mode-specific bitmap
  CX = width of bit map in bytes
  DX = height of bit map in pixel rows
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This call is ignored in text modes
  The image is drawn with its lower left corner at the current graphics
    cursor position
  The current clipping region is used, extended to a byte boundary
```

```
SeeAlso: AX=0011h,AX=0024h,AX=0034h,AX=0044h,AX=00A0h,AX=00A9h,AX=00B9h
--------V-620037-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_FONTSIZE" - SPECIFY FONT SIZE FOR TEXT OUTPUT
  AX = 0037h
  BX = desired character height in scan lines (8, 14, 16)
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This call is ignored if the current mode is not a VGA or SVGA graphics
    mode, or the specified character height is not 8, 14, or 16
SeeAlso: AX=0096h
--------V-620038-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_FREEPAGE" - FREE VIRTUAL OR LOGICAL VIDEO PAGE
  AX = 0038h
  BX = page number (0000h-003Fh)
Return: AX = status (0000h,0001h,0007h,0009h) (see #03473)
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0000h,AX=0001h,AX=0002h,AX=0003h
--------V-620039-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_GETADDR" - GET SEGMENT OF ACTIVE VIDEO PAGE
  AX = 0039h
Return: AX = segment of active video page
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0042h,AX=00ACh
--------V-62003A-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_GETATTR" - GET CHARACTER ATTRIB FOR POSITION
  AX = 003Ah
  BX = row
  CX = column
Return: AX = character attribute at specified location on active video page
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This call is ignored in graphics modes
SeeAlso: AX=000Bh,AX=003Dh,AX=0040h,AX=00A6h
--------V-62003B-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_GETBANKS" - GET SVGA READ AND WRITE BANKS
  AX = 003Bh
  ES:BX -> GETBANKS variable pointer record (see #03481)
Return: nothing
Note: the FGDRIVER functions are rearranged with each major release, but
```

```
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=00A7h

Format of FGDRIVER GETBANKS variable pointer record:
Offset  Size  Description (Table 03481)
 00h  WORD  segment of WORD buffer for read bank number
 02h  WORD  offset of WORD buffer for read bank number
 04h  WORD  segment of WORD buffer for write bank number
 06h  WORD  offset of WORD buffer for write bank number
--------V-62003C-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_GETBLOCK" - GRAB RECTANGLE OF DISPLAY
  AX = 003Ch
  ES:BX -> buffer for screen contents
  CX = leftmost column
  DX = rightmost column
  SI = top row
  DI = bottom row
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  In text modes, coordinates are character positions; in graphics modes,
    they are defined in screen space, and the left and right edges are
    adjusted to a byte boundary if necessary
  Use "FG_IMAGESIZ" (see AX=005Ch) to determine the required buffer size
SeeAlso: AX=0044h,AX=005Ch,AX=0098h
--------V-62003D-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_GETCHAR" - GET CHARACTER FOR SCREEN POSITION
  AX = 003Dh
  BX = row
  CX = column
Return: AX = character at specified location on active video page
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This call is ignored in graphics modes
SeeAlso: AX=000Bh,AX=003Ah,AX=0096h,AX=00A6h,AX=00C8h
--------V-62003E-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_GETCLIP" - GET CLIPPING REGION IN SCREEN SPACE
  AX = 003Eh
  ES:BX -> GETCLIP variable pointer record (see #03482)
Return: variables specifed by GETCLIP structure updated
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
```

```
SeeAlso: AX=00CAh

Format of FGDRIVER GETCLIP variable pointer record:
Offset  Size  Description (Table 03482)
 00h  WORD  segment of WORD buffer for minimum X coordinate (left edge)
 02h  WORD  offset of WORD buffer for minimum X coordinate (left edge)
 04h  WORD  segment of WORD buffer for maximum X coordinate (right edge)
 06h  WORD  offset of WORD buffer for maximum X coordinate (right edge)
 08h  WORD  segment of WORD buffer for minimum Y coordinate (top edge)
 0Ah  WORD  offset of WORD buffer for minimum Y coordinate (top edge)
 0Ch  WORD  segment of WORD buffer for maximum Y coordinate (bottom edge)
 0Eh  WORD  offset of WORD buffer for maximum Y coordinate (bottom edge)
--------V-62003F-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_GETCLOCK" - GET CLOCK TICKS SINCE MIDNIGHT
  AX = 003Fh
Return: DX:AX = number of clock ticks since midnight
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: INT 1A/AH=00h
--------V-620040-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_GETCOLOR" - GET CURRENT TEXT ATTRIBUTE
  AX = 0040h
Return: AX = current text attribute or color index (graphics modes)
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=000Bh,AX=003Ah,AX=00A6h
--------V-620041-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_GETDACS" - GET VIDEO DAC CONTENTS
  AX = 0041h
  CX = number of DAC registers to return (0001h to 0100h)
  DX = starting DAC register number (0000h to 00FFh)
  ES:BX -> buffer for DAC red/green/blue triples
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  The register number wraps back to zero after FFh
SeeAlso: AX=004Eh,AX=0086h,AX=00ABh
--------V-620042-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_GETENTRY" - GET PAGE TYPE AND ADDRESS
  AX = 0042h
  ES:BX -> variable pointer record (see #03483)
  CX = page number (00h-3Fh)
```

```
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  Page addresses are segments for pages stored in conventional memory
    (including physical and virtual pages), and EMS or XMS handles for
    pages stored in EMS or XMS
SeeAlso: AX=0000h,AX=00ACh

Format of FGDRIVER GETENTRY variable pointer record:
Offset  Size  Description (Table 03483)
 00h  WORD  segment of WORD buffer for page address
 02h  WORD  offset of WORD buffer for page address
 04h  WORD  segment of WORD buffer for page type (se #03484)
 06h  WORD  offset of WORD buffer for page type

(Table 03484)
Values for FGDRIVER page type:
 0000h  unallocated
 0001h  physical
 0002h  virtual
 0003h  logical page, stored in expanded memory (EMS)
 0004h  logical page, stored in extended memory (XMS)
 0005h  logical page, stored in conventional memory
--------V-620043-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_GETHPAGE" - GET CURRENT HIDDEN VIDEO PAGE NUM
  AX = 0043h
Return: AX = current hidden video page number (0000h-003Fh)
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=004Ch,AX=0050h,AX=00AEh
--------V-620044-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_GETIMAGE" - STORE IMAGE AS BITMAP
  AX = 0044h
  ES:BX -> buffer for video mode-specific bitmap
  CX = width of bitmap in bytes
  DX = height of bitmap in pixel rows
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0011h,AX=0024h,AX=0036h,AX=0099h,AX=00A0h
--------V-620045-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_GETINDEX" - GET COLOR VALUE FOR COLOR INDEX
  AX = 0045h
```

```
  BX = color index (0000h to 00FFh)
Return: AX = color value for specified color index
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This call returns the value passed to it in text and 256-color
    graphics modes
SeeAlso: AX=0018h,AX=0040h
--------V-620046-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_GETKEY" - GET NEXT KEYSTROKE
  AX = 0046h
  ES:BX -> variable pointer record (see #03485)
Return: (after next keystroke if no typeahead) variables updated
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=000Ah,AX=0063h,AX=0069h,AX=00DEh,INT 16/AH=00h

Format of FGDRIVER GETKEY variable pointer record:
Offset  Size  Description (Table 03485)
 00h  WORD  segment of BYTE buffer for ASCII keycode
 02h  WORD  offset of BYTE buffer for ASCII keycode
 04h  WORD  segment of BYTE buffer for extended keycode
 06h  WORD  offset of BYTE buffer for extended keycode
--------V-620047-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_GETLINES" - GET TEXT ROWS FOR CURR VIDEO MODE
  AX = 0047h
Return: AX = number of text rows on screen in current video mode
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=00AFh,AX=00C2h
--------V-620048-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_GETMAP" - STORE IMAGE AS MODE-INDEPENDNT BITMAP
  AX = 0048h
  ES:BX -> buffer for video mode-independent bitmap
  CX = width of bitmap in bytes
  DX = height of bitmap in pixel rows
Return: each bit in bitmap is set if corresponding pixel is of the current
    color, cleared otherwise
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This call is ignored in text modes
SeeAlso: AX=0011h,AX=001Eh,AX=0024h
```

```
--------V-620049-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_GETMAXX" - GET MAXIMUM COLUMN IN SCREEN SPACE
  AX = 0049h
Return: AX = maximum X coordinate in screen space
      (or character space if in text mode)
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=004Ah,AX=00C1h
--------V-62004A-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_GETMAXY" - GET MAXIMUM ROW IN SCREEN SPACE
  AX = 004Ah
Return: AX = maximum Y coordinate in screen space
    (or character space if in text mode)
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0049h,AX=00C1h
--------V-62004B-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_GETMODE" - GET CURRENT VIDEO MODE NUMBER
  AX = 004Bh
  ES:BX -> WORD ???
Return: AX = current video mode number
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=00B0h
--------V-62004C-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_GETPAGE" - GET ACTIVE VIDEO PAGE NUMBER
  AX = 004Ch
Return: AX = active video page (0000h-003Fh)
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0039h,AX=0042h,AX=0043h,AX=0050h,AX=00B2h
--------V-62004D-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_GETPIXEL" - GET COLOR OF SPECIFIED PIXEL
  AX = 004Dh
  BX = column in screen space
  CX = row in screen space
Return: AX = color value of pixel (0 to num_colors-1)
      0000h in text modes
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=003Ah,AX=003Dh,AX=008Fh,AX=0090h
```

```
--------V-62004E-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_GETRGB" - GET VIDEO DAC REGISTER CONTENTS
  AX = 004Eh
  ES:BX -> variable pointer record (see #03486)
  CX = DAC register number
Return: variables updated
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This call is ignored in text modes and CGA/EGA video modes (since
    they do not use DAC registers)
SeeAlso: AX=0041h,AX=0086h,AX=00ABh

Format of FGDRIVER GETRGB variable pointer record:
Offset  Size  Description (Table 03486)
 00h  WORD  segment of WORD buffer for red component of DAC register
 02h  WORD  offset of WORD buffer for red component
 04h  WORD  segment of WORD buffer for green component of DAC register
 06h  WORD  offset of WORD buffer for green component
 08h  WORD  segment of WORD buffer for blue component of DAC register
 0Ah  WORD  offset of WORD buffer for blue component
--------V-62004F-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_GETVIEW" - GET VIEWPORT EXTREME LIMITS
  AX = 004Fh
  ES:BX -> variable pointer record (see #03487)
Return: indicated variables updated
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=00B5h

Format of FGDRIVER GETVIEW variable pointer record:
Offset  Size  Description (Table 03487)
 00h  WORD  segment of WORD buffer for viewport left edge (viewport units)
 02h  WORD  offset of WORD buffer for viewport left edge (viewport units)
 04h  WORD  segment of WORD buffer for viewport right edge (viewport units)
 06h  WORD  offset of WORD buffer for viewport right edge (viewport units)
 08h  WORD  segment of WORD buffer for viewport top edge (viewport units)
 0Ah  WORD  offset of WORD buffer for viewport top edge (viewport units)
 0Ch  WORD  segment of WORD buffer for viewport bottom edge (viewp. units)
 0Eh  WORD  offset of WORD buffer for viewport bottom edge (viewp. units)
 10h  WORD  segment of WORD buffer for viewport left edge (screen space)
 12h  WORD  offset of WORD buffer for viewport left edge (screen space)
```

```
 14h  WORD  segment of WORD buffer for viewport right edge (screen space)
 16h  WORD  offset of WORD buffer for viewport right edge (screen space)
 18h  WORD  segment of WORD buffer for viewport top edge (screen space)
 1Ah  WORD  offset of WORD buffer for viewport top edge (screen space)
 1Ch  WORD  segment of WORD buffer for viewport bottom edge (screen space)
 1Eh  WORD  offset of WORD buffer for viewport bottom edge (screen space)
--------V-620050-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_GETVPAGE" - GET VISIBLE VIDEO PAGE NUMBER
  AX = 0050h
Return: AX = visible video page (0000h-003Fh)
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0043h,AX=004Ch,AX=00B6h
--------V-620051-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_GETXBOX" - GET WIDTH OF VERTICAL BOX LINES
  AX = 0051h
Return: AX = width (in pixels) of left and right edges of rectangles
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0006h,AX=0007h,AX=0008h,AX=006Bh
--------V-620052-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_GETXJOY" - GET HORIZONTAL POSITION OF JOYSTICK
  AX = 0052h
  BX = joystick number (0001h or 0002h)
Return: AX = horizontal position of joystick
      FFFFh if joystick uninitialized or not present
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  The actual coordinates are processor- and joystick-dependent
  You must call "FG_INITJOY" (AX=005Eh) before using this function
SeeAlso: AX=0009h,AX=0056h,AX=005Eh,AX=0062h,AX=0077h
--------V-620053-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_GETXJUST" - GET HORIZONTAL JUSTIFICATION
  AX = 0053h
Return: AX = string justification
      0000h strings are centered around current graphics X position
      0001h strings are right-justified at current graphics X position
      FFFFh strings are left-justified at current graphics X position
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=006Ch,AX=007Bh
```

```
--------V-620054-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_GETXPOS" - GET GRAPHICS CURSOR COLUMN
  AX = 0054h
Return: AX = screen space X coordinate of graphics cursor position
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0058h,AX=007Eh,AX=007Fh
--------V-620055-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_GETYBOX" - GET WIDTH OF HORIZONTAL BOX LINES
  AX = 0055h
Return: AX = width (in pixels) of top and bottom edges of rectangles
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0006h,AX=0007h,AX=0008h,AX=0063h
--------V-620056-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_GETYJOY" - GET VERTICAL POSITION OF JOYSTICK
  AX = 0056h
  BX = joystick number (0001h or 0002h)
Return: AX = vertical position of joystick
      FFFFh if joystick uninitialized or not present
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  The actual coordinates are processor- and joystick-dependent
  You must call "FG_INITJOY" (AX=005Eh) before using this function
SeeAlso: AX=0009h,AX=0052h,AX=005Eh,AX=0062h,AX=0077h
--------V-620057-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_GETYJUST" - GET VERTICAL JUSTIFICATION
  AX = 0057h
Return: AX = string justification
      0000h strings are centered around current graphics Y position
      0001h strings have top edge at current graphics Y position
      FFFFh strings have bottom edge at current graphics Y position
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0053h,AX=0065h
--------V-620058-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_GETYPOS" - GET GRAPHICS CURSOR ROW
  AX = 0058h
Return: AX = screen space Y coordinate of graphics cursor position
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
```

```
SeeAlso: AX=0054h,AX=007Eh,AX=007Fh
--------V-620059-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_HUSH" - STOP ASYNCHRONOUS SOUND IMMEDIATELY
  AX = 0059h
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This function immediately stops any sounds started with "FG_MUSICB",
    "FG_SOUNDS", or "FG_VOICES"; it is ignored if no asynchronous sound
    is playing
SeeAlso: AX=005Ah,AX=0081h,AX=00BDh,AX=00DCh
--------V-62005A-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_HUSHNEXT" - STOP ASYNCHRONOUS SOUND
  AX = 005Ah
Program: FGDRIVER is the external video driver for the shareware
    Fastgraph/Light by Ted Gruber Software
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This function stops any sounds started with "FG_MUSICB", "FG_SOUNDS",
    or "FG_VOICES" after the current repetition completes; it is ignored
    unless asynchronous sound is continuous
SeeAlso: AX=0059h,AX=0081h,AX=00BDh,AX=00DCh
--------V-62005B-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_IMAGEBUF" - SPECIFY TEMPORARY IMAGE BUFFER
  AX = 005Bh
  ES:BX -> buffer to be used when creating or displaying GIF/PCX/PPR/SPR
      images
  CX = size of buffer in bytes or 0000h to use internal buffer
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  Fastgraph's internal buffer is 4096 bytes; this function allows the
    application to define a larger buffer which will typically speed
    processing
SeeAlso: AX=006Ch,AX=006Dh,AX=006Eh,AX=00B9h,AX=00BAh,AX=00BBh
--------V-62005C-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_IMAGESIZ" - DETERMINE IMAGE STORAGE REQUIREMENT
  AX = 005Ch
  BX = image width in pixels
  CX = image height in pixels
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
Return: DX:AX = size in bytes of mode-specific bitmap for current video mode
```

```
SeeAlso: AX=003Ch,AX=0098h
--------V-62005D-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_INITEMS" - INITIALIZE EXPANDED MEMORY USE
  AX = 005Dh
Return: AX = status
      0000h successful
      FFFFh expanded memory manager inaccessible or not installed
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0002h,AX=0060h"v4.02"
--------V-62005E-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_INITJOY" - INITIALIZE JOYSTICK USE
  AX = 005Eh
  BX = joystick number (0001h or 0002h)
Return: AX = status
      0000h successful
      FFFFh joystick not connected or no game port
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  Fastgraph assumes that the requested joystick is centered at the time
    this function is called
SeeAlso: AX=0009h,AX=0052h,AX=0056h,AX=0062h,AX=0077h
--------V-62005F-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_INITPM"??? - (NOT IMPLEMENTED)
  AX = 005Fh
  ???
Return: ???
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
--------V-620060-----------------------------
INT 62 u - FGDRIVER v1.10 only - "FG_DISPPCX" - DISPLAY PCX FILE
  AX = 0060h
  ES:BX -> ASCIZ filename
  CX = flags
      bit 0: use current palette rather than PCX file's palette
      bits 1-15 reserved (0)
Return: AX = status
      0000h success
      0001h file not found
      0002h file is not a valid PCX file
Notes:  the FGDRIVER functions are rearranged with each major release, but
```

```
    their parameters do not change (see #03474 at AX=0000h)
  This call is ignored in text modes and Hercules low-resolution graphics
  The image is displayed with its upper left corner at the current
    graphics cursor position
SeeAlso: AX=00B9h
--------V-620060-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_INITXMS" - INITIALIZE EXTENDED MEMORY USE
  AX = 0060h
Return: AX = status
      0000h successful
      FFFFh extended memory manager inaccessible or not installed
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0003h,AX=005Dh
--------V-620061-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_INSIDE" - CHECK IF POINT INSIDE CONVEX POLYGON
  AX = 0061h
  ES:BX -> vertex array
  CX = number of vertices
  DX = screen space column of point to test
  SI = screen space row of point to test
Return: AX = result
      0000h point is outside polygon
      0001h point is inside polygon
      undefined if not a convex polygon
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0092h,AX=0093h,AX=0094h
--------V-620062-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_INTJOY" - GET KEYCODES CORRESP TO JOYSTICK POS
  AX = 0062h
  CX = joystick number (0001h or 0002h)
  ES:BX -> variable pointer record (see #03488)
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  If the indicated joystick has not been initialized with AX=008Eh, both
    the button code and joystick position will be set to 00h
  If either button is pressed, a button code of 0Dh is returned;
    otherwise, a button code of 00h is returned
SeeAlso: AX=0009h,AX=0052h,AX=0046h,AX=005Eh,AX=0077h
```

```
Format of FGDRIVER INTJOY variable pointer record:
Offset  Size  Description (Table 03488)
 00h  WORD  segment of BYTE buffer for button code
 02h  WORD  offset of BYTE buffer for button code
 04h  WORD  segment of BYTE buffer for joystick position
 06h  WORD  offset of BYTE buffer for joystick position
--------V-620063-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_INTKEY" - GET KEYSTROKE, NO WAIT
  AX = 0063h
  ES:BX -> variable pointer record (see #03489)
Return: variables updated
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  If the keyboard buffer is empty, both the ASCII and extended keycodes
    are set to 00h
SeeAlso: AX=000Ah,AX=0046h,AX=0069h,AX=00DEh,INT 16/AH=01h

Format of FGDRIVER INTKEY variable pointer record:
Offset  Size  Description (Table 03489)
 00h  WORD  segment of BYTE buffer for ASCII keycode
 02h  WORD  offset of BYTE buffer for ASCII keycode
 04h  WORD  segment of BYTE buffer for extended keycode
 06h  WORD  offset of BYTE buffer for extended keycode
--------V-620064-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_INVERT" - INVERT ORIENTATION OF BITMAP
  AX = 0064h
  ES:BX -> bitmap
  CX = width of bitmap in bytes
  DX = height of bitmap in pixel rows
Return: nothing
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
--------V-620065-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_JUSTIFY" - SET TEXT OUTPUT JUSTIFICATION
  AX = 0065h
  BX = horizontal justification
        (00h centered, 01h right-justified, FFh left-justified)
  CX = vertical justification
        (00h centered, 01h top of characters, FFh bottom of chars)
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
```

```
SeeAlso: AX=000Ch,AX=0037h,AX=0053h,AX=0057h,AX=0096h
--------V-620066----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_KBINIT" - ENABLE/DISABLE LOW-LEVEL KBD HANDLER
  AX = 0066h
  BX = new state (0000h disabled, 0001h enabled)
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  When the low-level handler is enabled, "FG_GETKEY", "FG_INTKEY",
    "FG_WAITKEY", and all other functions calling DOS or BIOS keyboard
    services become unavailable
SeeAlso: AX=0067h,AX=0068h,AX=0069h
--------V-620067----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_KBLAST" - GET MOST RECENT SCANCODE PROCESSED
  AX = 0067h
Return: AX = scancode for keypress most recently processed by FastGraph's
        low-level keyboard handler, or 0000h if no keys since FG_KBINIT
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0066h,AX=0068h,AX=0069h
--------V-620068----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_KBRESET" - RESET LOW-LEVEL KEYBOARD HANDLER
  AX = 0068h
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0066h,AX=0067h,AX=0069h
--------V-620069----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_KBTEST" - CHECK WHETHER SPECIFIED KEY PRESSED
  AX = 0069h
  BX = scan code of desired key
Return: AX = state
      0000h key is not pressed
      0001h key is currently pressed
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  The low-level keyboard handler must have been enabled with AX=0066h
SeeAlso: AX=0066h,AX=0067h,INT 16/AH=01h
--------V-62006A----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_LOADPCX" - LOAD .PCX INTO ACTIVE VIRTUAL BUFFER
  AX = 006Ah
  ES:BX -> ASCIZ filename for PCX image
  CX = control flags
```

```
      bit 0: use current palette, overriding stored .PCX palette
      bit 1: load at current graphics position, not stored position
      bit 2: load image from FG_IMAGEBUF buffer instead of .PCX file
Return: AX = status
      0000h successful
      0001h file not found
      0002h not a .PCX file
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=006Ch,AX=00B9h,AX=00D5h
--------V-62006B-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_LOCATE" - SET TEXT-MODE CURSOR POSITION
  AX = 006Bh
  BX = row
  CX = column
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  There are only eight text cursors shared by successive groups of
    eight video pages (pages 0, 8, 16, ... share one cursor, 1, 9, ...
    share the second, etc)
SeeAlso: AX=0054h,AX=0058h,AX=007Eh,AX=00E0h
--------V-62006C-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_MAKEPCX" - CREATE PCX FILE FROM SCREEN WINDOW
  AX = 006Ch
  BX = left edge in screen space units
  CX = right edge in screen space units
  DX = top edge in screen space units
  SI = bottom edge in screen space units
  ES:DI -> ASCIZ filename of PCX file to create
Return: AX = status
      0000h successful
      0001h file not created
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  The PCX file is created from the specified region of the active video
    page; the left and right edges are adjusted to a byte boundary if
    necessary
  If the specified file already exists, it is overwritten
  This call is ignored in text and Hercules low-resolution graphics modes
SeeAlso: AX=005Bh,AX=006Dh,AX=006Ah,AX=006Eh,AX=00B9h,AX=00BAh,AX=00BBh
--------V-62006D-----------------------------
```

```
INT 62 u - FGDRIVER v4.02 - "FG_MAKEPPR" - CREATE PACKED PIXEL RUN FILE
  AX = 006Dh
  BX = left edge in screen space units
  CX = right edge in screen space units
  DX = top edge in screen space units
  SI = bottom edge in screen space units
  ES:DI -> ASCIZ filename of PPR file to create
Return: AX = status
      0000h successful
      0001h file not created
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  The PPR file is created from the specified region of the active video
    page; the left and right edges are adjusted to byte boundaries if
    necessary
  If the specified file already exists, it is overwritten
  This call is ignored in text modes
SeeAlso: AX=005Bh,AX=006Ch,AX=006Eh,AX=00B9h,AX=00BAh,AX=00BBh
--------V-62006E-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_MAKESPR" - CREATE STANDARD PIXEL RUN FILE
  AX = 006Eh
  BX = left edge in screen space units
  CX = right edge in screen space units
  DX = top edge in screen space units
  SI = bottom edge in screen space units
  ES:DI -> ASCIZ filename of SPR file to create
Return: AX = status
      0000h successful
      0001h file not created
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  The SPR file is created from the specified region of the active video
    page; the left and right edges are adjusted to byte boundaries if
    necessary
  If the specified file already exists, it is overwritten
  This call is ignored in text modes
SeeAlso: AX=005Bh,AX=006Ch,AX=006Dh,AX=00B9h,AX=00BAh,AX=00BBh
--------V-62006F-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_MAPRGB" - MAP COLOR COMPONENTS INTO PALETTE VAL
  AX = 006Fh
  BX = red component
```

```
  CX = green component
  DX = blue component
Return: AX = mode-specific palette value corresponding to specified components
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This function is only meaningful in 16-color graphics modes
SeeAlso: AX=004Eh,AX=00B3h
--------V-620070-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_MEASURE" - GET DELAY UNITS PER CLOCK TICK
  AX = 0070h
Return: AX = delay units per clock tick (processor-dependent)
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  These delay units are used by "FG_STALL" (see AX=00BFh)
SeeAlso: AX=00BFh
--------V-620071-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_MEMAVAIL" - QUERY AMOUNT OF FREE MEMORY
  AX = 0071h
Return: DX:AX = number of bytes of conventional memory available
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=005Dh,AX=0072h
--------V-620072-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_MEMORY" - GET SIZE OF VIDEO MEMORY
  AX = 0072h
Return: AX = size of video memory in KB
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  You must first call "FG_SVGAINIT" (see AX=00C1h) before using this
    function
SeeAlso: AX=0071h
--------V-620073-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_MOUSE256" - DEFINE 256-COLOR MOUSE CURSOR
  AX = 0073h
  ES:BX -> 512-byte cursor definition
  CX = X offset of hot-spot within mouse cursor
  DX = Y offset of hot-spot within mouse cursor
Return: nothing
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  the cursor definition array consists of 256 bytes defining the 16x16
```

```
    screen mask, followed by another 256 bytes defininig the 16x16
    cursor mask
SeeAlso: AX=0074h,AX=0075h,AX=007Bh
--------V-620074----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_MOUSEBUT" - GET MOUSE BUTTON PRESS/RELEASE CNTS
  AX = 0074h
  CX = mouse button (1 = left press, 2 = right press, 3 = middle press,
      -1=left release, -2=right release, -3=middle release)
  ES:BX -> variable pointer record (see #03490)
Return: variables updated
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This function returns the count of presses or releases since the last
    call to this function; if the count is zero, row and column will
    also be zero
SeeAlso: AX=0075h,AX=0077h,AX=007Ah

Format of FGDRIVER MOUSEBUT variable pointer record:
Offset  Size  Description (Table 03490)
 00h  WORD  segment of WORD buffer for press/release count
 02h  WORD  offset of WORD buffer for press/release count
 04h  WORD  segment of WORD buffer for screen space col of last press/rls
 06h  WORD  offset of WORD buffer for screen space col of last press/rels
 08h  WORD  segment of WORD buffer for screen space row of last press/rls
 0Ah  WORD  offset of WORD buffer for screen space row of last press/rls
--------V-620075----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_MOUSECUR" - SPECIFY TEXT-MODE MOUSE CURSOR
  AX = 0075h
  BX = screen mask
  CX = cursor mask
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This call is ignored in graphics modes
SeeAlso: AX=007Bh,AX=007Dh
--------V-620076----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_MOUSEFIN" - UNHOOK FASTGRAPH MOUSE HANDLER
  AX = 0076h
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This function should be called before switching back to text mode if
    "FG_MOUSEINI" was called while in an SVGA graphics mode
```

```
SeeAlso: AX=0077h
--------V-620077-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_MOUSEINI" - INITIALIZE MOUSE SUPPORT
  AX = 0077h
Return: AX = status
      0002h two-button mouse
      0003h three-button mouse
      FFFFh initialization failed
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  After this call, the mouse cursor is invisible
SeeAlso: AX=0076h,AX=0078h,AX=007Ch
--------V-620078-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_MOUSELIM" - SPECIFY MOUSE CURSOR LIMITS
  AX = 0078h
  BX = left-most position allowed for mouse cursor
  CX = right-most position allowed
  DX = top-most position allowed
  SI = bottom-most position allowed
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0077h,AX=0079h,INT 33/AX=0007h,INT 33/AX=0008h
--------V-620079-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_MOUSEMOV" - SET MOUSE CURSOR POSITION
  AX = 0079h
  BX = new column
  CX = new row
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This function will not move the mouse cursor outside the bounding box
    specified with AX=009Bh
SeeAlso: AX=0078h,AX=007Ah,INT 33/AX=0004h
--------V-62007A-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_MOUSEPOS" - GET CURRENT MOUSE POSITION
  AX = 007Ah
  ES:BX -> variable pointer record (see #03491)
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0079h

Format of FGDRIVER MOUSEPOS variable pointer record:
```

```
Offset  Size  Description (Table 03491)
 00h  WORD  segment of WORD buffer for mouse column
 02h  WORD  offset of WORD buffer for mouse column
 04h  WORD  segment of WORD buffer for mouse row
 06h  WORD  offset of WORD buffer for mouse row
 08h  WORD  segment of WORD buffer for button status
 0Ah  WORD  offset of WORD buffer for button status
Note: button status: bit 0 = left button, bit 1 = right, bit 2 = middle
--------V-62007B-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_MOUSEPTR" - SPECIFY GRAPH-MODE MOUSE CURSOR
  AX = 007Bh
  ES:BX -> masks (16-byte screen mask followed by 16-byte cursor mask)
  CX = X offset of hot spot from upper left corner
  DX = Y offset of hot spot from upper left corner
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This call is ignored in text modes
SeeAlso: AX=0073h,AX=0075h,INT 33/AX=0009h
--------V-62007C-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_MOUSESPD" - SET MOUSE CURSOR SPEED
  AX = 007Ch
  BX = horizontal mickeys per eight pixels of movement (default 16)
  CX = vertical mickeys per eight pixels of movement (default 16)
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: INT 33/AX=000Fh,INT 33/AX=001Ah
--------V-62007D-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_MOUSEVIS" - SET MOUSE CURSOR VISIBILITY
  AX = 007Dh
  BX = new state (0000h invisible, 0001h visible)
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0077h,INT 33/AX=0001h,INT 33/AX=0002h
--------V-62007E-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_MOVE" - SET GRAPHICS CURSOR POSITION
  AX = 007Eh
  BX = new column
  CX = new row
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This call is ignored in text modes
```

```
SeeAlso: AX=0054h,AX=0058h,AX=006Bh,AX=0079h,AX=007Fh,AX=00E0h
--------V-62007F-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_MOVEREL" - ADJUST GRAPHICS CURSOR POSITION
  AX = 007Fh
  BX = column offset
  CX = row offset
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This call is ignored in text modes
SeeAlso: AX=0054h,AX=0058h,AX=007Eh
--------V-620080-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_MUSIC" - PLAY SERIES OF NOTES
  AX = 0080h
  ES:BX -> '$'-terminated music string in BASIC PLAY format
Return: after music completed
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This call is ignored if asynchronous sound (AX=00A5h,AX=00A8h,
    AX=00AAh) is already in progress
SeeAlso: AX=0081h,AX=00BCh,AX=00DBh
--------V-620081-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_MUSICB" - PLAY SERIES OF NOTES IN BACKGROUND
  AX = 0081h
  CX = number of repetitions (negative = continuous play)
  ES:BX -> '$'-terminated music string in BASIC PLAY format
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This call is ignored if asynchronous sound (AX=00A5h,AX=00A8h,
    AX=00AAh) is already in progress
SeeAlso: AX=0059h,AX=0081h,AX=00BDh,AX=00DCh
--------V-620082-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_NUMLOCK" - GET STATE OF NUMLOCK KEY
  AX = 0082h
Return: AX = NumLock state (0000h off, 0001h on)
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=000Ah,AX=00A4h,AX=00A8h,AX=00B1h
--------V-620083-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_PACK" - CONVERT PIXEL-PER-BYTE TO PACKED BITMAP
  AX = 0083h
  ES:BX -> variable pointer record (see #03492)
```

```
  CX = width of bitmap in pixels
  DX = height of bitmap in pixels
Return: result bitmap buffer updated
SeeAlso: AX=00CBh

Format of FGDRIVER PACK variable pointer record:
Offset  Size  Description (Table 03492)
 00h  WORD  segment of source (mode-independent) bitmap
 02h  WORD  offset of source (mode-independent) bitmap
 04h  WORD  segment of buffer for result (mode-specific) bitmap
 06h  WORD  offset of buffer for result (mode-specific) bitmap
--------V-620084-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_PAGESIZE" - GET VIDEO PAGE SIZE FOR CURR MODE
  AX = 0084h
Return: DX:AX = page size in bytes
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0072h
--------V-620085-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_PAINT" - FLOOD CLOSED REGION WITH COLOR
  AX = 0085h
  BX = column
  CX = row
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This function fills an arbitrary closed region around the specified
    point with the current color; the screen edges are not considered
    region boundaries
  This call is ignored in text modes
SeeAlso: AX=0035h
--------V-620086-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_PALETTE" - SET PALETTE / SET VIDEO DAC REGISTER
  AX = 0086h
---CGA 4-color graphics---
  BX = CGA paletee number
  CX = background color
---CGA 2-color graphics---
  BX ignored
  CX = foreground color
---16-color graphics---
  BX = palette register number
```

```
  CX = palette value
---256-color graphics---
  BX = DAC register number
  CX =  DAC value
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This function is ignored in text modes and Hercules graphics modes
  Few EGA/VGA adapters correctly set the foreground color in CGA mode 6
SeeAlso: AX=0041h,AX=004Eh,AX=0087h,AX=00ABh
--------V-620087-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_PALETTES" - SET ALL PALETTE REGISTERS
  AX = 0087h
  ES:BX -> array of 16 WORDs containing values for palette registers
      (or first 16 DAC registers in 256-color modes)
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This function is ignored in text modes, CGA and Hercules graphics modes
SeeAlso: AX=0041h,AX=004Eh,AX=0086h,AX=00ABh
--------V-620088-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_PAN" - SET SCREEN ORIGIN
  AX = 0088h
  BX = new column for screen origin
  CX = new row for screen origin
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=00B2h
--------V-620089-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_PATTERN" - SPECIFY DISPLAY PATTERN FOR COLOR
  AX = 0089h
  BX = index of pattern to define
  CX = number of predefined display pattern
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  When displaying a pixel run map, Fastgraph uses the pattern associated
    with each color index rather than displaying the actual color
  This call has no effect in text and 256-color graphics modes
SeeAlso: AX=0023h,AX=0045h
--------V-62008A-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_PCXHEAD" - GET PCX FILE HEADER
  AX = 008Ah
  ES:BX -> variable pointer record (see #03493)
```

```
Return: AX = status
      0000h successful
      FFFEh not a PCX file
      FFFFh file does not exist
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=008Bh,AX=00B9h

Format of FGDRIVER PCXHEAD variable pointer record:
Offset  Size  Description (Table 03493)
 00h  WORD  segment of ASCIZ filename
 02h  WORD  offset of ASCIZ filename
 04h  WORD  segment of 128-byte buffer for PCX header
 06h  WORD  offset of 128-byte buffer for PCX header
--------V-62008B-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_PCXMODE" - GET OPTIMAL VIDEO MODE FOR DISPLAY
  AX = 008Bh
  ES:BX -> PCX header (see AX=008Ah)
Return: AX = optimal video mode for PCX file
      FFFEh not a valid PCX header
      FFFFh unable to determine compatible video mode
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=008Ah,AX=00B9h
--------V-62008C-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_PCXPAL" - GET PALETTE STORED IN PCX FILE
  AX = 008Ch
  ES:BX -> variable pointer record (see #03494)
Return: AX = number of colors in palette (16 or 256) or
      FFFEh not a valid PCX file
      FFFFh file not found
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=008Dh

Format of FGDRIVER PCXPAL variable pointer record:
Offset  Size  Description (Table 03494)
 00h  WORD  segment of ASCIZ .PCX filename
 02h  WORD  offset of ASCIZ .PCX filename
 04h  WORD  segment of buffer for .PCX palette RGB triples
 06h  WORD  offset of buffer for .PCX palette RGB triples
```

```
Note: the buffer for the palette must hold at least three times as many bytes
    as there are colors in the palette
SeeAlso: #03495
--------V-62008D-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_PCXRANGE" - GET EXTENT OF PCX IMAGE
  AX = 008Dh
  ES:BX -> variable pointer record (see #03495)
Return: indicated variables updated; if the indicated header is not valid,
    all of the coordinate variables are set to FFFFh
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=008Ch

Format of FGDRIVER PCXRANGE variable pointer record:
Offset  Size  Description (Table 03495)
 00h  WORD  segment of 128-byte PCX file header
 02h  WORD  offset of 128-byte PCX file header
 04h  WORD  segment of WORD buffer for X coordinate of image's left edge
 04h  WORD  offset of WORD buffer for X coordinate of image's left edge
 08h  WORD  segment of WORD buffer for X coordinate of image's right edge
 0Ah  WORD  offset of WORD buffer for X coordinate of image's right edge
 0Ch  WORD  segment of WORD buffer for Y coordinate of image's top edge
 0Eh  WORD  offset of WORD buffer for Y coordinate of image's top edge
 10h  WORD  segment of WORD buffer for X coordinate of image's bottom edge
 12h  WORD  offset of WORD buffer for X coordinate of image's bottom edge
SeeAlso: #03494
--------V-62008E-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_PLAYING" - DETERMINE WHETHER ASYNC SOUND ACTIVE
  AX = 008Eh
Return: AX = sound state (0 = no asynchronous sound, 1 = async sound playing)
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0059h,AX=0081h,AX=00BDh,AX=00DCh
--------V-62008F-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_POINT" - DISPLAY A PIXEL
  AX = 008Fh
  BX = column
  CX = row
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This call is ignored in text modes
```

```
SeeAlso: AX=0006h,AX=001Dh,AX=004Dh,AX=0090h
--------V-620090-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_POINTX" - DISPLAY A PIXEL IN XOR MODE
  AX = 0090h
  BX = column
  CX = row
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This call is ignored in text modes
SeeAlso: AX=004Dh,AX=008Fh
--------V-620091-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_POLYEDGE" - SPECIFY INCLUSION OF FINAL PIXELS
  AX = 0091h
  BX = edge flag
       0000h include right- and bottom-edge pixels when drawing polygons
       with FG_POLYFILL
       0001h (default) exclue right- and bottom-edge pixels
SeeAlso: AX=0092h
--------V-620092-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_POLYFILL" - DRAW A FILLED CONVEX POLYGON
  AX = 0092h
  CX = number of vertices
  ES:BX -> variable pointer record (see #03496)
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  The vertex array consists of pairs of words specifying the X and Y
    coordinates of each vertex; the work array is used internally and
    must contain at least four times as many bytes as the polygon is
    high in pixels
  This function is ignored in text modes
  If the polygon is non-convex, only a portion of it may be filled
SeeAlso: AX=000Eh,AX=0027h,AX=0091h,AX=0093h,AX=0094h,AX=0095h

Format of FGDRIVER POLYFILL variable pointer record:
Offset  Size  Description (Table 03496)
 00h  WORD  segment of vertex array (see #03498)
 02h  WORD  offset of vertex array
 04h  WORD  segment of work array
 06h  WORD  offset of work array
SeeAlso: #03497
--------V-620093-----------------------------
```

```
INT 62 u - FGDRIVER v4.02 - "FG_POLYGON" - DRAW AN UNFILLED POLYGON
  AX = 0093h
  CX = number of vertices in polygon
  ES:BX -> variable pointer record (see #03497)
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This call is ignored in text modes
SeeAlso: AX=0091h,AX=0092h,AX=0094h,AX=0095h

Format of FGDRIVER POLYGON variable pointer record:
Offset  Size  Description (Table 03497)
 00h  WORD  segment of WORD array containing vertex columns
 02h  WORD  offset of WORD array containing vertex columns
 04h  WORD  segment of WORD array containing vertex rows
 06h  WORD  offset of WORD array containing vertex rows
SeeAlso: #03496
--------V-620094-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_POLYLINE" - DRAW AN UNFILLED POLYGON
  AX = 0094h
  CX = number of vertices in polygon
  ES:BX -> vertex array (see #03498)
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This function is equivalent to "FG_POLYGON", but uses only a single
    array to define the vertices
  This call is ignored in text modes
SeeAlso: AX=008Fh,AX=0092h,AX=0093h,AX=0095h

Format of FGDRIVER POLYFILL/POLYLINE vertex array element:
Offset  Size  Description (Table 03498)
 00h  WORD  column
 02h  WORD  row
SeeAlso: #03497
--------V-620095-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_POLYOFF" - DEFINE POLYGON DRAWING OFFSET
  AX = 0095h
  BX = horizontal offset (default 0)
  CX = vertical offset (default 0)
Desc: define the offsets applied to all vertices of polygons drawn with
    "FG_POLYFILL" or "FG_POLYLINE"
Note: the FGDRIVER functions are rearranged with each major release, but
```

```
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0092h,AX=0094h
--------V-620096-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_PRINT" - DISPLAY STRING OF HARDWARE CHARACTERS
  AX = 0096h
  CX = length of string
  ES:BX -> string to display
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  The string is displayed in the current color, and the graphics cursor
    is updated to be just to the right of the last displayed character
  This function is ignored in text modes
SeeAlso: AX=000Ch,AX=0037h,AX=0065h,AX=0097h
--------V-620097-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_PRINTC" - DISPLAY STRING OF HW CHARS (CLIPPED)
  AX = 0097h
  CX = length of string
  ES:BX -> string to display
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  The string is displayed in the current color, and the graphics cursor
    is updated to be just to the right of the last displayed character;
    only the portion of the string that lies within the current clipping
    area is displayed
  This function is ignored in text modes
SeeAlso: AX=0096h
--------V-620098-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_PUTBLOCK" - RESTORE RECTANGLE OF DISPLAY
  AX = 0098h
  ES:BX -> buffer containing previously-saved image
  CX = left edge
  DX = right edge
  SI = top edge
  DI = bottom edge
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  In text modes, coordinates are character positions; in graphics modes,
    they are defined in screen space, and the left and right edges are
    adjusted to a byte boundary if necessary
SeeAlso: AX=003Ch,AX=0099h
--------V-620099-----------------------------
```

```
INT 62 u - FGDRIVER v4.02 - "FG_PUTIMAGE" - DISPLAY MODE-SPECIFIC BITMAP IMAGE
  AX = 0099h
  ES:BX -> buffer containing mode-specific bitmap
  CX = width in bytes
  DX = height in pixel rows
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  The bitmap is displayed with its lower left corner at the graphics
    cursor position; color 0 is NOT treated as transparent
SeeAlso: AX=0011h,AX=0024h,AX=0036h,AX=0044h,AX=0098h
--------V-62009A-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_QUIET" - STOP CONTINUOUS SYNCHRONOUS SOUND
  AX = 009Ah
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This call has no effect if there is no continuous sound playing
SeeAlso: AX=0059h,AX=0080h,AX=00BDh,AX=00DCh
--------V-62009B-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_RECT" - DRAW UNFILLED RECTANGLE IN SCREEN SPACE
  AX = 009Bh
  BX = left edge column
  CX = right edge column
  DX = top edge row
  SI = bottom edge row
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0006h,AX=0012h,AX=0023h,AX=003Ch
--------V-62009C-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_RESET" - ERASE SCREEN AND RESTORE SCREEN ATTR
  AX = 009Ch
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This call is ignored in graphics modes
  The screen attributes are only restored if ANSI.SYS is loaded
SeeAlso: AX=0028h,AX=00B0h
--------V-62009D-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_RESIZE" - SET GRAPHICS MODE VIDEO PAGE SIZE
  AX = 009Dh
  BX = new page width in pixels
  CX = new page height in pixels
Notes:  the FGDRIVER functions are rearranged with each major release, but
```

```
    their parameters do not change (see #03474 at AX=0000h)
  The visible page must be set to 0000h before making this call
  The mouse, joysticks, expanded memory, and extended memory must be
    reinitialized after this call
SeeAlso: AX=0084h,AX=00ACh
--------V-62009E----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_RESTORE" - COPY REGION FROM HIDDEN TO VIS PAGE
  AX = 009Eh
  BX = left edge column
  CX = right edge column
  DX = top edge row
  SI = bottom edge row
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  The left and right edges are adjusted to byte boundaries if necessary
SeeAlso: AX=00A2h,AX=00C6h
--------V-62009F----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_RESUME" - RESTART ASYNCHRONOUS SOUND
  AX = 009Fh
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=00BDh,AX=00C0h
--------V-6200A0----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_REVIMAGE" - DISPLAY REVERSED IMAGE (BITMAP)
  AX = 00A0h
  ES:BX -> mode-specific bitmap
  CX = width of bitmap in bytes
  DX = height of bitmap in pixel rows
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This call is ignored in text modes
  The image is drawn with its lower left corner at the current graphics
    cursor position
SeeAlso: AX=0011h,AX=0024h,AX=0036h,AX=0044h,AX=00A1h,AX=00A9h,AX=00B9h
--------V-6200A1----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_REVMASK" - DISPLAY REVERSED IMAGE (MASKING MAP)
  AX = 00A1h
  ES:BX -> array containing image stored as a masking map (see #03475)
  CX = number of pixel runs in masking map
  DX = width of masking map in pixels
Notes:  the FGDRIVER functions are rearranged with each major release, but
```

```
    their parameters do not change (see #03474 at AX=0000h)
  This call is ignored in text modes and in native EGA and VGA graphics
    modes
  The image is drawn with its lower left corner at the current graphics
    cursor position
SeeAlso: AX=0010h,AX=001Fh,AX=0034h,AX=00A0h,AX=00A9h,AX=00C5h
--------V-6200A2-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_SAVE" - COPY REGION FROM VISIBLE TO HIDDEN PAGE
  AX = 00A2h
  BX = left edge column
  CX = right edge column
  DX = top edge row
  SI = bottom edge row
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  The left and right edges are adjusted to byte boundaries if necessary
SeeAlso: AX=009Eh,AX=00C6h
--------V-6200A3-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_SCALE" - SCALE BITMAP
  AX = 00A3h
  ES:BX -> variable pointer record (see #03499)
  CX = width of source bitmap in pixels (0 < width <= 1024)
  DX = height of source bitmap in pixels (0 < height <= 1024)
  SI = desired width of scaled bitmap (0 < width <= 1024)
  DI = desired height of scaled bitmap (0 < height <= 1024)
Return: nothing
SeeAlso: AX=00B7h

Format of FGDRIVER SCALE variable pointer record:
Offset  Size  Description (Table 03499)
 00h  WORD  segment of source bitmap (one pixel per byte)
 02h  WORD  offset of source bitmap
 04h  WORD  segment of buffer for resulting scaled bitmap
 06h  WORD  offset of buffer for resulting scaled bitmap
--------V-6200A4-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_SCRLOCK" - GET STATE OF SCROLL LOCK KEY
  AX = 00A4h
Return: AX = ScrollLock state (0000h off, 0001h on)
Program: FGDRIVER is the external video driver for the shareware
    Fastgraph/Light by Ted Gruber Software
Note: the FGDRIVER functions are rearranged with each major release, but
```

```
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=000Ah,AX=0082h,AX=00A8h,AX=00B1h
--------V-6200A5-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_SCROLL" - VERTICALLY SCROLL SCREEN REGION
  AX = 00A5h
  BX = left edge column
  CX = right edge column
  DX = top edge row
  SI = bottom edge row
  DI = number of pixels by which to scroll (positive scrolls up,
       negative scrolls down)
  ES = type of scroll
      0000h circular (rows scrolled off are copied to vacated rows)
      else vacated rows are filled with the current color
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  In graphics modes, the left and right edges are adjusted to byte
    boundaries if necessary
  Circular scrolling uses part of the hidden page as a workspace
SeeAlso: AX=0088h,AX=00B2h
--------V-6200A6-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_SETATTR" - SET TEXT-MODE CHARACTER ATTRIBUTE
  AX = 00A6h
  BX = foreground
  CX = background
  DX = blink (0000h nonblinking, 0001h blink)
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This call is ignored in graphics modes
SeeAlso: AX=000Bh,AX=003Ah,AX=0048h
--------V-6200A7-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_SETBANKS" - SET SVGA READ AND WRITE BANKS
  AX = 00A7h
  BX = memory bank from which to read (FFFFh leave unchanged)
  CX = memory bank to which to write (FFFFh leave unchanged)
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=003Bh
--------V-6200A8-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_SETCAPS" - SET STATE OF CAPSLOCK KEY
  AX = 00A8h
```

```
  BX = new state (0000h off, 0001h on)
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=000Ah,AX=0082h,AX=00A4h,AX=00B1h
--------V-6200A9-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_SETCLIP" - SET CLIPPING REGION
  AX = 00A9h
  BX = left edge of clipping region
  CX = right edge of clipping region
  DX = top edge of clipping region
  SI = bottom edge of clipping region
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0010h,AX=001Fh,AX=0034h,AX=003Eh,AX=00A1h,AX=00C5h
--------V-6200AA-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_SETCOLOR" - SET CURRENT COLOR
  AX = 00AAh
  BX = new color index (or text attribute in text modes)
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0018h,AX=0040h,AX=0045h
--------V-6200AB-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_SETDACS" - SET VIDEO DAC CONTENTS
  AX = 00ABh
  CX = number of DAC registers to set (0001h to 0100h)
  DX = starting DAC register number (0000h to 00FFh)
  ES:BX -> buffer containing DAC red/green/blue triples
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  The register number wraps back to zero after reaching FFh
  This call has no effect in text modes or graphics modes below 11h
SeeAlso: AX=0041h,AX=004Eh,AX=0086h,INT 10/AX=1012h
--------V-6200AC-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_SETENTRY" - SET TYPE AND ADDRESS OF VIDEO PAGE
  AX = 00ACh
  BX = page number (00h-3Fh)
  CX = page address
  DX = page type (see #03484 at AX=0042h)
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0038h,AX=0042h,AX=00B0h
```

```
--------V-6200AD-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_SETFUNC" - SET LOGICAL OPERATION FOR VIDEO OPS
  AX = 00ADh
  BX = operation
       0000h replacement
       0001h AND
       0002h OR
       0003h XOR
Notes:  the FGDRIVER functions are rearranged with each major release, but
     their parameters do not change (see #03474 at AX=0000h)
  This function is only available in native EGA/VGA graphics modes
     (0Dh to 12h)
SeeAlso: AX=001Eh,AX=008Fh
--------V-6200AE-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_SETHPAGE" - SET HIDDEN VIDEO PAGE
  AX = 00AEh
  BX = new hidden page (0000h to 003Fh)
Notes:  the FGDRIVER functions are rearranged with each major release, but
     their parameters do not change (see #03474 at AX=0000h)
  The specified page must be a physical page or a virtual page
SeeAlso: AX=0043h,AX=00B2h,AX=00B6h
--------V-6200AF-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_SETLINES" - SET TEXT ROWS ON SCREEN
  AX = 00AFh
  BX = new screen size (25, 43, 50)
Note: the FGDRIVER functions are rearranged with each major release, but
     their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0047h
--------V-6200B0-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_SETMODE" - SELECT VIDEO MODE AND INITIALIZE
  AX = 00B0h
  BX = new video mode or FFFFh for current mode (see #03500)
  ES:DX -> WORD shareware splash screen flag
Notes:  the FGDRIVER functions are rearranged with each major release, but
     their parameters do not change (see #03474 at AX=0000h)
  This call resets the active video page to page 0000h, the clipping
     region to the entire screen, text rows to 25, etc.
  in the shareware version, FGDRIVER sets the word pointed at by ES:DX
     to 9090h after it shows the shareware announcement to the user
SeeAlso: AX=004Bh,INT 10/AH=00h
```

```
(Table 03500)
Values for FGDRIVER video mode:
 00h-07h standard BIOS modes
 09h  PCjr/Tandy1000 320x200x16
 0Bh  Hercules graphics 720x348
 0Ch  Hercules graphics 320x200
 0Dh-13h standard BIOS modes
 14h  VGA graphics 320x200x256
 15h  VGA graphics 320x400x256
 16h  VGA graphics 320x240x256
 17h  VGA graphics 320x480x256
 18h  SVGA graphics 640x400x256
 19h  SVGA graphics 640x480x256
 1Ah  SVGA graphics 800x600x256
 1Bh  SVGA graphics 1024x768x256
 1Ch  SVGA graphics 800x600x16
 1Dh  SVGA graphics 1024x768x16
--------V-6200B1-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_SETNUM" - SET STATE OF NUMLOCK KEY
  AX = 00B1h
  BX = new state (0000h off, 0001h on)
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=000Ah,AX=0082h,AX=00A4h,AX=00A8h
--------V-6200B2-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_SETPAGE" - SET ACTIVE VIDEO PAGE
  AX = 00B2h
  BX = new video page (0000h to 003Fh)
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  The specified page must be a physical or virtual page
SeeAlso: AX=0039h,AX=0042h,AX=0043h,AX=0050h,AX=0088h
--------V-6200B3-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_SETRGB" - SET VIDEO DAC REGISTER CONTENTS
  AX = 00B3h
  BX = palette or DAC register number
  CX = red color component
  DX = green component
  SI = blue component
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
```

```
  The register number may be negative for Tandy, PCjr, and 200-line
    EGA graphics modes to specify an intense color
  This call has no effect in text, CGA graphics, and Hercules graphics
    modes
SeeAlso: AX=004Eh
--------V-6200B4----------------------------
INT 62 u - FGDRIVER v4.02 - UNUSED
  AX = 00B4h
Return: AX = 0000h
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
--------V-6200B5----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_SETVIEW" - DEFINE VIEWPORT
  AX = 00B5h
  ES:BX -> variable record (see #03501)
  CX = viewport's top edge in screen space
  DX = viewport's bottom edge in screen space
Return: nothing
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
SeeAlso: AX=004Fh

Format of FGDRIVER SETVIEW variable record:
Offset  Size  Description (Table 03501)
 00h  WORD  viewport's left edge in viewport units
 02h  WORD  viewport's right edge in viewport units
 04h  WORD  viewport's top edge in viewport units
 06h  WORD  viewport's bottom edge in viewport units
 08h  WORD  viewport's left edge in screen space
 0Ah  WORD  viewport's right edge in screen space
--------V-6200B6----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_SETVPAGE" - SET VISIBLE VIDEO PAGE
  AX = 00B6h
  BX = new video page (0000h to 003Fh)
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  The specified page must be a physical or virtual page
SeeAlso: AX=0043h,AX=00AEh,AX=00B2h
--------V-6200B7----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_SHEAR" - SHEAR UNPACKED BITMAP
  AX = 00B7h
```

```
  ES:BX -> variable pointer record (see #03502)
  CX = bitmap width (0 < width <= 1024)
  DX = bitmap height (0 < height <= 1024)
  SI = size of resulting image (width for horiz. shear, height for vert.)
  DI = shear type
       0000h horizontal shear to left
       0001h horizontal shear to right
       0002h vertical shear to left (left edge stretched up)
       0003h vertical shear to right (right edge stretched up)
Return: nothing
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
SeeAlso: AX=00A3h

Format of FGDRIVER SHEAR variable pointer record:
Offset  Size  Description (Table 03502)
 00h  WORD  segment of source bitmap
 02h  WORD  offset of source bitmap
 04h  WORD  segment of buffer for resulting bitmap
 06h  WORD  offset of buffer for resulting bitmap
--------V-6200B8-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_SHOWFLIC" - DISPLAY IMAGE FROM FLI/FLC FILE
  AX = 00B8h
  ES:BX -> ASCIZ filename for FLI/FLC file
  CX = number of times to display image file (0000h = continuously)
  DX = control flags (see #03479 at AX=0031h)
Return: AX = status
      0000h successful
      0001h file not found
      0002h not a valid FLI/FLC file
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
  the FLI/FLC display may be stopped by the user by pressing Esc
SeeAlso: AX=0030h,AX=0031h,AX=00B9h
--------V-6200B9-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_SHOWPCX" - DISPLAY IMAGE FROM PCX FILE
  AX = 00B9h
  ES:BX -> ASCIZ filename of PCX image
  CX = flags (see #03503)
Return: AX = status
      0000h success
```

```
      0001h file not found
      0002h not a PCX file
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This function is ignored in text modes and Hercules low-res graphics
SeeAlso: AX=005Bh,AX=0060h"1.10",AX=006Ch,AX=006Dh,AX=006Eh,AX=00B8h,AX=00BAh
SeeAlso: AX=00BBh

Bitfields for FGDRIVER flags:
Bit(s)  Description (Table 03503)
 0  use current palette rather than PCX file's palette
 1  display image at cursor position instead of position in PCX header
 2-15 reserved
--------V-6200BA-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_SHOWPPR" - DISPLAY IMAGE FROM PPR FILE
  AX = 00BAh
  ES:BX -> ASCIZ filename of packed pixel run image
  CX = width in pixels (nonzero)
Return: AX = status
      0000h successful
      0001h file not found
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  The image is displayed with its lower left corner at the current
    graphics cursor position
  This function is ignored in text modes and 256-color graphics modes
SeeAlso: AX=005Bh,AX=006Ch,AX=006Dh,AX=006Eh,AX=00B9h,AX=00BBh
--------V-6200BB-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_SHOWSPR" - DISPLAY IMAGE FROM SPR FILE
  AX = 00BBh
  ES:BX -> ASCIZ filename of standard pixel run image
  CX = width in pixels (nonzero)
Return: AX = status
      0000h successful
      0001h file not found
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  The image is displayed with its lower left corner at the current
    graphics cursor position
  This function is ignored in text modes
SeeAlso: AX=005Bh,AX=006Ch,AX=006Dh,AX=006Eh,AX=00B9h,AX=00BAh
```

```
--------V-6200BC-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_SOUND" - MAKE SOUND FOR SPECIFIED DURATION
  AX = 00BCh
  BX = frequency in Hertz (18-32767)
  CX = duration in clock ticks (0000h or negative for continuous sound)
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This call is ignored if asynchronous sound (AX=00A5h,AX=00A8h,
    AX=00AAh) is already in progress
SeeAlso: AX=0059h,AX=0080h,AX=00BDh,AX=00C0h,AX=00DBh
--------V-6200BD-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_SOUNDS" - PLAY SOUNDS IN BACKGROUND
  AX = 00BDh
  CX = number of times to cycle through sound list
  ES:BX -> sounds array (see #03504)
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This call is ignored if asynchronous sound (AX=00A5h,AX=00A8h,
    AX=00AAh) is already in progress
SeeAlso: AX=0059h,AX=0081h,AX=00BCh,AX=00C0h,AX=00DCh

Format of FGDRIVER sounds array element:
Offset  Size  Description (Table 03504)
 00h  WORD  frequency of sound in Hertz (0000h ends array)
 02h  WORD  duration of sound in clock ticks
--------V-6200BE-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_SPLIT" - ENABLE/DISABLE SPLIT SCREEN ENVIRONMNT
  AX = 00BEh
  BX = beginning row for bottom half of split-screen
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
--------V-6200BF-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_STALL" - PAUSE FOR SPECIFIED DURATION
  AX = 00BFh
  BX = duration in processor-dependent delay units (see AX=0070h)
Return: after delay elapses
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0070h,INT 2F/AX=1224h
--------V-6200C0-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_SUSPEND" - TEMPORARILY STOP ASYNCHRONOUS SOUND
```

```
  AX = 00C0h
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This call has no effect if there is no asynchronous sound in progress
  The program must not exit while sound is suspended
SeeAlso: AX=0059h,AX=0080h,AX=00BDh,AX=00C0h,AX=00DBh
--------V-6200C1-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_SVGAINIT" - INITIALIZE FASTGRAPH SVGA KERNEL
  AX = 00C1h
  BX = method (see #03505)
Return: AX = status
      0000h no VESA BIOS or supported SVGA chipset
      0001h using VESA BIOS
      0002h-0016h specific chipset being used (same as "method" below)
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This function must be called before attempting to set SVGA graphics
    modes (18h to 1Dh) or using "FG_BESTMODE", "FG_TESTMODE", or
    "FG_MEMORY"
SeeAlso: AX=0025h,AX=00C2h,AX=00C3h

(Table 03505)
Values for FGDRIVER SVGA method:
 0000h  autodetect, give chipset-specific code priority over VESA
 0001h  autodetect, give VESA priority over chipset-specific code
 0002h  Ahead type "A"
 0003h  Ahead type "B"
 0004h  ATI 18800
 0005h  ATI 18800-1
 0006h  ATI 28800
 0007h  Chips & Technologies 82c451/455/456
 0008h  C&T 82c452
 0009h  C&T 82c453
 000Ah  Genoa 6000 series
 000Bh  Oak OTI-067
 000Ch  Paradise PVGA1a
 000Dh  Paradise WD90C00/WD90C10
 000Eh  Paradise WD90C11/WD90C30/WD90C31
 000Fh  Trident 8800
 0010h  Trident 8900
 0011h  Tseng ET3000
```

```
 0012h  Tseng ET4000
 0013h  Video7
 0014h  Cirrus Logic 5400 series
 0015h  S3
 0016h  Trident 8900B/8900C/9000
--------V-6200C2-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_SVGASTAT" - GET SVGA CHIPSET INFORMATION
  AX = 00C2h
Return: AX = chipset information (see #03506)
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0025h,AX=00C1h,AX=00C3h

Bitfields for FGDRIVER chipset information:
Bit(s)  Description (Table 03506)
 0  SVGA kernel initialized
 1  VESA support enabled
 2  extended video pages available in modes 13-23
 3  SVGA chipset has separate read and write banks
 4-15 reserved (0)
--------V-6200C3-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_SVGAVER" - GET FASTGRAPH SVGA KERNEL VERSION
  AX = 00C3h
  ES:BX -> variable pointer record (see #03507)
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=00C1h,AX=00C2h

Format of FGDRIVER variable pointer record:
Offset  Size  Description (Table 03507)
 00h  WORD  segment of WORD buffer for major version
 02h  WORD  offset of WORD buffer for major version
 04h  WORD  segment of WORD buffer for minor version (hundredths)
 06h  WORD  offset of WORD buffer for minor version
--------V-6200C4-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_TCDEFINE" - DEFINE TRANSPARENCY OF COLOR INDEX
  AX = 00C4h
  BX = color index
  CX = transparency (00h opaque, other transparent)
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
```

```
  This function is ignored in text modes
SeeAlso: AX=00C5h,AX=00C6h
--------V-6200C5----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_TCMASK" - SET TRANSPARENT COLORS
  AX = 00C5h
  BX = colors to consider transparent (bit 0 = color 0, etc)
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This call is ignored in text modes
  The specified colors are considered transparent by "FG_TCXFER"
SeeAlso: AX=00C4h,AX=00C6h
--------V-6200C6----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_TCXFER" - COPY REGION EXCLUDING TRANSPARENT
  AX = 00C6h
  CX = source video page
  DX = destination video page
  ES:BX -> copy record (see #03508)
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  Pixels which are in any of the colors defined as transparent with
    "FG_TCMASK" (see AX=00C5h) are left unchanged in the destination
    region
  The source and destination regions must not overlap if they are located
    on the same page
  This call is ignored in text modes
SeeAlso: AX=00C4h,AX=00C5h,AX=00CAh
--------V-6200C7----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_TESTMODE" - CHECK IF VIDEO MODE AVAILABLE
  AX = 00C7h
  BX = desired video mode (00h-17h, also 18h-1Dh after "FG_SVGAINIT")
  CX = required number of video pages (ignore memory size if <= 0)
Return: AX = status
      0000h mode not available with requested number of pages
      0001h mode is available
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0005h,AX=00B0h
--------V-6200C8----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_TEXT" - DISPLAY STRING OF CHARACTERS
  AX = 00C8h
  CX = length of string
```

```
  ES:BX -> string
Notes:  the FGDRIVER functions are rearranged with each major release, but
     their parameters do not change (see #03474 at AX=0000h)
  The string is displayed starting at the text cursor position using the
     current text attribute (text modes) or color index (graphics modes)
  The text cursor position is updated after this call
SeeAlso: AX=000Ch,AX=0096h,AX=00C9h
--------V-6200C9-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_TEXTC" - DISPLAY STRING OF CHARACTERS (CLIPPED)
  AX = 00C9h
  CX = length of string
  ES:BX -> string
Notes:  the FGDRIVER functions are rearranged with each major release, but
     their parameters do not change (see #03474)
  The string is displayed starting at the text cursor position using the
     current text attribute (text modes) or color index (graphics modes),
     showing only the portion within the current clipping window
  The text cursor position is updated after this call
SeeAlso: AX=00C8h
--------V-6200CA-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_TRANSFER" - COPY REGION
  AX = 00CAh
  CX = source video page
  DX = destination video page
  ES:BX -> copy record (see #03508)
Notes:  the FGDRIVER functions are rearranged with each major release, but
     their parameters do not change (see #03474 at AX=0000h)
  The source and destination regions must not overlap if they are located
     on the same page
SeeAlso: AX=009Eh,AX=00A2h,AX=00A5h,AX=00C6h

Format of FGDRIVER copy record:
Offset  Size  Description (Table 03508)
 00h  WORD  left edge column of source region
 02h  WORD  right edge column of source region
 04h  WORD  top edge row of source region
 06h  WORD  bottom edge row of source region
 08h  WORD  left edge of destination
 0Ah  WORD  bottom edge of destination
--------V-6200CB-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_UNPACK" - EXPAND MODE-SPECIFIC BITMAP
```

```
  AX = 00CBh
  ES:BX -> variable pointer record (see #03509)
  CX = size of source bitmap in bytes
Return: result buffer filled
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
SeeAlso: AX=0083h

Format of FGDRIVER UNPACK variable pointer record:
Offset  Size  Description (Table 03509)
 00h  WORD  segment of source bitmap
 02h  WORD  offset of source bitmap
 04h  WORD  segment of buffer for resulting bitmap
 06h  WORD  offset of buffer for resulting bitmap
--------V-6200CC-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_VBADDR" - GET ADDRESS OF VIRTUAL BUFFER
  AX = 00CCh
  BX = virtual buffer handle (0000h-001Fh)
Return: DX:AX -> virtual buffer
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
SeeAlso: AX=00CDh,AX=00CEh,AX=00CFh,AX=00D3h
--------V-6200CD-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_VBALLOC" - CREATE VIRTUAL BUFFER
  AX = 00CDh
  BX = width of virtual buffer in pixels
  CX = height in pixels
Return: AX = handle for virtual buffer, or
      FFFEh out of memory
      FFFFh virtual buffer table full
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
SeeAlso: AX=00CCh,AX=00CEh,AX=00D0h,AX=00D1h,AX=00D2h,AX=00D4h
--------V-6200CE-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_VBCLOSE" - CLOSE ACTIVE VIRTUAL BUFFER
  AX = 00CEh
Desc: close the active virtual buffer and direct further graphics to the
    active video page
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
SeeAlso: AX=00CCh,AX=00D1h,AX=00D4h,AX=00D5h,AX=00D9h
```

```
--------V-6200CF-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_VBCOPY" - COPY RECT BETWEEN VIRTUAL BUFFERS
  AX = 00CFh
  ES:BX -> variable record (see #03510)
  CX = handle for source virtual buffer
  DX = handle for destination virtual buffer
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
  if the destination buffer is the same as the source buffer, the regions
    must not overlap
SeeAlso: AX=00D0h,AX=00D1h,AX=00D4h,AX=00D6h,AX=00D7h

Format of FGDRIVER VBCOPY variable record:
Offset  Size  Description (Table 03510)
 00h  WORD  source region's left edge
 02h  WORD  source region's right edge
 04h  WORD  source region's top edge
 06h  WORD  source region's bottom edge
 08h  WORD  X coordinate of destination's upper left corner
 0Ah  WORD  Y coordinate of destination's upper left corner
--------V-6200D0-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_VBCUT" - COPY RECT FROM VIDEO TO VIRTUAL BUFFER
  AX = 00D0h
  BX = source region's left edge
  CX = source region's right edge
  DX = source region's top edge
  SI = source region's bottom edge
  DI = X coordinate of destination's upper left corner
  ES = Y coordinate of destination's upper left corner
Desc: copy a rectangle from the active video page to the active virtual
    buffer
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
SeeAlso: AX=00CDh,AX=00CFh,AX=00D4h,AX=00D6h
--------V-6200D1-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_VBDEFINE" - CREATE VIRTUAL BUFFER
  AX = 00D1h
  ES:BX -> memory block for virtual buffer
  CX = buffer width in pixels
  DX = buffer height in pixels
Return: AX = virtual buffer handle
```

```
      FFFFh on error
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
  the application-provided buffer must be large enough to hold CX*DX
    bytes
SeeAlso: AX=00CCh,AX=00CDh,AX=00CEh,AX=00D2h,AX=00D4h,AX=00D5h
--------V-6200D2----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_VBFREE" - RELEASE VIRTUAL BUFFER
  AX = 00D2h
  BX = virtual buffer handle
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
  this function should be used only for buffers created with FG_VBALLOC
SeeAlso: AX=00CDh,AX=00D4h,AX=00D5h
--------V-6200D3----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_VBHANDLE" - GET ACTIVE VIRTUAL BUFFER'S HANDLE
  AX = 00D3h
Return: AX = handle for active virtual buffer
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
SeeAlso: AX=00CCh,AX=00D4h
--------V-6200D4----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_VBINIT" - INITIALIZE VIRTUAL BUFFER ENVIRONMENT
  AX = 00D4h
Return: nothing
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
  this function must be called before any other virtual buffer functions
    are used
SeeAlso: AX=00CDh,AX=00D5h
--------V-6200D5----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_VBOPEN" - MAKE VIRTUAL BUFFER ACTIVE
  AX = 00D5h
  BX = virtual buffer handle
Return: AX = status
      0000h successful
      FFFEh no buffer defined for specified handle
      FFFFh invalid buffer handle
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
SeeAlso: AX=00CDh,AX=00CEh,AX=00D1h,AX=00D4h,AX=00D9h
```

```
--------V-6200D6-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_VBPASTE" - COPY RECT FROM VIRTUAL BUF TO VIDEO
  AX = 00D6h
  BX = source region's left edge
  CX = source region's right edge
  DX = source region's top edge
  SI = source region's bottom edge
  DI = X coordinate of destination's upper left corner
  ES = Y coordinate of destination's upper left corner
Desc: copy a rectangle from the active virtual buffer to the active video
    page
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
SeeAlso: AX=00CFh,AX=00D0h,AX=00D4h,AX=00D7h
--------V-6200D7-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_VBTCCOPY" - COPY RECTANGLE BETWEEN VIRTUAL BUFS
  AX = 00D7h
  ES:BX -> variable record (see #03511)
  CX = source virtual buffer's handle
  DX = destination virtual buffer's handle
Desc: copy rectangle from one virtual buffer to another with transparent
    colors
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
  if the destination buffer is the same as the source buffer, the regions
    must not overlap
SeeAlso: AX=00CFh,AX=00D0h,AX=00D6h,AX=00D8h

Format of FGDRIVER VBTCCOPY variable record:
Offset  Size  Description (Table 03511)
 00h  WORD  source region's left edge
 02h  WORD  source region's right edge
 04h  WORD  source region's top edge
 06h  WORD  source region's bottom edge
 08h  WORD  X coordinate of destination region's upper left corner
 0Ah  WORD  Y coordinate of destination region's upper left corner
--------V-6200D8-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_VBTCXFER" - COPY RECTANGLE TO ACTIVE VIDEO PAGE
  AX = 00D8h
  BX = source region's left edge
  CX = source region's right edge
```

```
  DX = source region's top edge
  SI = source region's bottom edge
  DI = X coordinate of destination's upper left corner
  ES = Y coordinate of destination's upper left corner
Desc: copy a rectangle from the active virtual buffer to the active video
    page, with transparent colors
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
SeeAlso: AX=00D6h,AX=00D7h
--------V-6200D9-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_VBUNDEF" - RELEASE HANDLE FOR VIRTUAL BUFFER
  AX = 00D9h
  BX = virtual buffer handle
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
  the specified handle must not reference the currently-active virtual
    buffer
SeeAlso: AX=00D1h,AX=00D5h
--------V-6200DA-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_VGASTATE" - SAVE/RESTORE VGA CONTROLLER STATE
  AX = 00DAh
  BX = direction (0000h save, else restore)
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474)
  this function is only meaningful in modes 0Dh and above
--------V-6200DB-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_VOICE" - START SOUND
  AX = 00DBh
  BX = channel on TI sound chip
      1-3 = channels 1-3, 4 = channel 4 with periodic noise,
      5 = channel 4 with white noise
  CX = frequency in Hz (18-32767 for channels 1-3; 0=512 Hz, 1=1024 Hz,
       2=2048 Hz for channels 4 and 5)
  DX = volume
  SI = duration in clock ticks (continuous if <= 0)
Program: FGDRIVER is the external video driver for the shareware
    Fastgraph/Light by Ted Gruber Software
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This function is only available on the PCjr and Tandy 1000 machines
SeeAlso: AX=0080h,AX=00BCh,AX=00DCh
```

```
--------V-6200DC-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_VOICES" - PLAY SOUNDS IN BACKGROUND
  AX = 00DCh
  ES:BX -> tone array (see #03512)
  CX = number of times to repeat tone array
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  This function is only available on the PCjr and Tandy 1000 machines
SeeAlso: AX=0059h,AX=0081h,AX=00BDh,AX=00DBh

Format of FGDRIVER tone array element:
Offset  Size  Description (Table 03512)
 00h  WORD  channel number (0000h terminates array)
 02h  WORD  frequency
 04h  WORD  volume
 06h  WORD  duration in 1/72.8 seconds
--------V-6200DD-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_WAITFOR" - DELAY FOR SPECIFIED DURATION
  AX = 00DDh
  BX = duration in clock ticks
Return: after delay elapses
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=00BFh,INT 1A/AX=FF01h
--------V-6200DE-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_WAITKEY" - FLUSH KEYBOARD BUFFER AND AWAIT KEY
  AX = 00DEh
Return: after next key pressed
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=000Ah,AX=0046h,AX=0063h,AX=0069h,INT 16/AH=00h
--------V-6200DF-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_WAITVR" - ENABLE/DISABLE VERTICAL RETRACE WAIT
  AX = 00DFh
  BX = new state (0000h disabled, 0001h enabled)
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
--------V-6200E0-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_WHERE" - GET CURRENT CURSOR POSITION
  AX = 00E0h
  ES:BX -> variable pointers (see #03513)
```

```
Return: indicated variables filled with cursor row and column for active
    display
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=0054h,AX=0058h,AX=006Bh,AX=007Eh

Format of FGDRIVER variable pointers:
Offset  Size  Description (Table 03513)
 00h  WORD  segment of WORD buffer for cursor row
 02h  WORD  offset of WORD buffer for cursor row
 04h  WORD  segment WORD buffer for cursor column
 06h  WORD  offset WORD buffer for cursor column
--------V-6200E1-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_XALPHA" - CONVERT SCREEN COLUMN TO CHAR COLUMN
  AX = 00E1h
  BX = screen space column
Return: AX = character space column containing specified coordinate
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=00E2h,AX=00E4h
--------V-6200E2-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_XCONVERT" - CONVERT CHAR COLUMN TO SCREEN COL
  AX = 00E2h
  BX = character space column
Return: AX = screen space column of leftmost pixel in specified character col
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=00E2h,AX=00E5h
--------V-6200E3-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_XVIEW" - CONVERT VIEWPORT COORDINATE
  AX = 00E3h
  BX = horizontal viewport coordinate
Return: AX = screen space X coordinate corresponding to supplied coordinate
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  if no viewport has been defined, the returned coordinate will be the
    same as the viewport coordinate
SeeAlso: AX=00E2h,AX=00E6h
--------V-6200E4-----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_YALPHA" - CONVERT SCREEN ROW TO CHARACTER ROW
  AX = 00E4h
```

```
  BX = screen space row
Return: AX = character space row containing specified coordinate
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=00E1h,AX=00E5h
--------V-6200E5----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_YCONVERT" - CONVERT CHARACTER ROW TO SCREEN ROW
  AX = 00E5h
  BX = character space row
Return: AX = screen space row of topmost pixel in specified character row
Note: the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
SeeAlso: AX=00E2h,AX=00E4h,AX=00E6h
--------V-6200E6----------------------------
INT 62 u - FGDRIVER v4.02 - "FG_YVIEW" - TRANSLATE VERTICAL VIEWPORT COORDINATE
  AX = 00E6h
  BX = viewport Y coordinate
Return: AX = screen space row for viewport coordinate
Program: FGDRIVER is the external video driver for the shareware
    Fastgraph/Light by Ted Gruber Software
Notes:  the FGDRIVER functions are rearranged with each major release, but
    their parameters do not change (see #03474 at AX=0000h)
  if no viewport has been defined, the returned coordinate will be the
    same as the viewport coordinate
SeeAlso: AX=00E3h,AX=00E5h
--------T-6201----------------------------
INT 62 - Cswitch - GIVE UP REST OF TIME-SLICE
  AH = 01h
Program: Cswitch is a set of multitasking functions by Herb Rose
SeeAlso: AH=05h"Cswitch",AH=06h"Cswitch",INT 15/AX=1000h,INT 2F/AX=1680h
--------N-6201----------------------------
INT 62 - BW-TCP - HW DRIVER (ETHDEV.SYS) - NOP for ETHDEV.ODI
  AH = 01h
Return: CF clear if successful
  CF set on error
     AL = error code
Range:  INT 4C to INT FB, selected by configuration
SeeAlso: AH=FEh,INT 21/AH=3Fh"BW-TCP",INT 63/AH=24h"BW-TCP"
SeeAlso: INT 64/AH=01h"BW-NFS"
--------T-6202----------------------------
INT 62 - Cswitch - WAIT FOR SEMAPHORE
```

```
  AH = 02h
  DX = semaphore number (0-63)
Return: AX = FFFFh bad semaphore number
       else  success
SeeAlso: AH=03h"Cswitch",AH=04h"Cswitch"
--------N-6202------------------------------
INT 62 - BW-TCP - HW DRIVER (ETHDEV.SYS) - INITIALIZE
  AH = 02h
Return: CF clear if successful
  CF set on error
      AL = error code
SeeAlso: AH=00h"ETHDEV",AH=03h"ETHDEV",AH=FEh,INT 21/AH=3Fh"BW-TCP"
SeeAlso: INT 63/AH=03h"BW-TCP",INT 63/AH=24h"BW-TCP",INT 64/AH=01h"BW-NFS"
--------T-6203------------------------------
INT 62 - Cswitch - CHECK SEMAPHORE
  AH = 03h
  DX = semaphore number (0-63)
Return: AX = status
      FFFFh not owned
      else  owned
SeeAlso: AH=02h,AH=04h
--------N-6203------------------------------
INT 62 - BW-TCP - HW DRIVER (ETHDEV.SYS) - GET REAL IP ADDRESS
  AH = 03h
  DS:SI -> DWORD buffer for IP address
Return: CF clear if successful
  CF set on error
      AL = error code
Range:  INT 4C to INT FB, selected by configuration
Note: the Beame&Whiteside TCP/IP protocol stack uses two consecutive
    interrupts (62h and 63h by default); the BW-NFS client uses a third
    consecutive interrupt (64h by default) if it is loaded
SeeAlso: AH=FEh,INT 21/AH=3Fh"BW-TCP",INT 63/AH=24h"BW-TCP"
SeeAlso: INT 64/AH=01h"BW-NFS"
--------T-6204------------------------------
INT 62 - Cswitch - TRIGGER SEMAPHORE
  AH = 04h
  DX = semaphore number (0-63)
Return: AX = status
      FFFFh bad semaphore number
      else  success
```

```
SeeAlso: AH=02h"Cswitch",AH=03h"Cswitch"
--------N-6204-------------------------------
INT 62 - BW-TCP - HW DRIVER (ETHDEV.SYS) - SET ???
  AH = 04h
  BX = ???
  ES:SI -> FAR routine for ???
Return: CF clear if successful
  CF set on error
      AL = error code
SeeAlso: AH=FEh,INT 21/AH=3Fh"BW-TCP",INT 63/AH=24h"BW-TCP"
SeeAlso: INT 64/AH=01h"BW-NFS"
--------T-6205-------------------------------
INT 62 - Cswitch - SLEEP
  AH = 05h
  BX = seconds to sleep
SeeAlso: AH=01h"Cswitch",AH=06h"Cswitch",AH=08h"Cswitch"
--------N-6205-------------------------------
INT 62 - BW-TCP - HW DRIVER (ETHDEV.SYS) - ???
  AH = 05h
  ???
Return: CF clear if successful
  CF set on error
      AL = error code
SeeAlso: AH=FEh,INT 21/AH=3Fh"BW-TCP",INT 63/AH=24h"BW-TCP"
SeeAlso: INT 64/AH=01h"BW-NFS"
--------T-6206-------------------------------
INT 62 - Cswitch - SUSPEND
  AH = 06h
SeeAlso: AH=05h"Cswitch",AH=08h"Cswitch"
--------N-6206-------------------------------
INT 62 - BW-TCP - HW DRIVER (ETHDEV.SYS) - ???
  AH = 06h
  ???
Return: CF clear if successful
  CF set on error
      AL = error code
Range:  INT 4C to INT FB, selected by configuration
Note: the Beame&Whiteside TCP/IP protocol stack uses two consecutive
    interrupts (62h and 63h by default); the BW-NFS client uses a third
    consecutive interrupt (64h by default) if it is loaded
SeeAlso: AH=FEh,INT 21/AH=3Fh"BW-TCP",INT 63/AH=24h"BW-TCP"
```

```
SeeAlso: INT 64/AH=01h"BW-NFS"
--------T-6207-------------------------------
INT 62 - Cswitch - SPAWN
  AH = 07h
  ES:BX -> function address to start executing at
  CX = priority (1-10)
Return: AX = result/status
      FFFDh  no free memory control blocks
      FFFEh  no free task control blocks
      FFFFh  not enough memory to create new task stack
      >0     the tcb number of the new task, indicating no error
SeeAlso: AH=0Fh"Cswitch",AH=10h"Cswitch"
--------N-6207-------------------------------
INT 62 - BW-TCP - HW DRIVER (ETHDEV.SYS) - ???
  AH = 07h
  DS:SI -> ???
Return: CF clear if successful
  CF set on error
      AL = error code
SeeAlso: AH=FEh,INT 21/AH=3Fh"BW-TCP",INT 63/AH=24h"BW-TCP"
SeeAlso: INT 64/AH=01h"BW-NFS"
--------T-6208-------------------------------
INT 62 - Cswitch - WAKE UP TASK
  AH = 08h
  BX = tcb identifier
SeeAlso: AH=05h,AH=06h
--------N-6208-------------------------------
INT 62 - BW-TCP - HW DRIVER (ETHDEV.SYS) - ???
  AH = 08h
  CX = ???
  ES:SI -> ??? buffer (see #03514)
Return: CF clear if successful
  CF set on error
      AL = error code
Range:  INT 4C to INT FB, selected by configuration
SeeAlso: AH=FEh,INT 21/AH=3Fh"BW-TCP",INT 63/AH=24h"BW-TCP"
SeeAlso: INT 64/AH=01h"BW-NFS"

Format of BW-TCP ??? buffer:
Offset  Size  Description (Table 03514)
 00h  6 BYTEs hardware address???
```

```
 06h  6 BYTEs ???
 0Ch  WORD  ???
 0Eh  WORD  ???
----------6208--CXFFFE----------------------
INT 62 - MS SQL Server/Sybase DBLIBRARY interface - UNINSTALL/GET PSP ADDR
  AH = 08h
  CX = FFFEh
  DX = FFFFh
Return: AX = PSP address of resident DBLIBRARY
Note: this call does not free the memory allocated to the TSR; the calling
    code must do the deallocation.
SeeAlso: INT 62"DBLIBRARY"
--------T-6209------------------------------
INT 62 - Cswitch - SET PRIORITY
  AH = 09h
  BX = new base priority (1-10)
Note: the lower the priority is numerically, the more often the task will run
--------N-6209------------------------------
INT 62 - BW-TCP - HW DRIVER (ETHDEV.SYS) - HOOK TIMER INTERRUPT
  AH = 09h
Return: CF clear if successful
      AX = handler ID
  CF set on error
      AL = error code
Range:  INT 4C to INT FB, selected by configuration
Note: the Beame&Whiteside TCP/IP protocol stack uses two consecutive
    interrupts (62h and 63h by default); the BW-NFS client uses a third
    consecutive interrupt (64h by default) if it is loaded
SeeAlso: AH=0Ah"ETHDEV",AH=FEh,INT 21/AH=3Fh"BW-TCP",INT 63/AH=24h"BW-TCP"
SeeAlso: INT 64/AH=01h"BW-NFS"
--------T-620A------------------------------
INT 62 - Cswitch - TEST MESSAGE QUEUE
  AH = 0Ah
  DX = queue number (0-63)
Return: AX = result/message size
      0000h nothing on queue
      FFFFh bad queue number
      else  number of bytes in first message in queue
SeeAlso: AH=0Bh"Cswitch",AH=0Ch"Cswitch"
--------N-620A------------------------------
INT 62 - BW-TCP - HW DRIVER (ETHDEV.SYS) - UNHOOK TIMER INTERRUPT
```

```
  AH = 0Ah
  DX = handler ID
Return: CF clear if successful
  CF set on error
      AL = error code
SeeAlso: AH=09h"ETHDEV",AH=FEh,INT 21/AH=3Fh"BW-TCP",INT 63/AH=24h"BW-TCP"
SeeAlso: INT 64/AH=01h"BW-NFS"
--------T-620B--------------------------------
INT 62 - Cswitch - SEND MESSAGE
  AH = 0Bh
  CX = number of bytes to write
  DS:SI -> buffer
  DX = queue number (0-63)
Return: AX = result/message size
      0000h no message was on queue
      FFFEh triggered by something arriving, redo the call
      FFFFh bad queue number
      else  number of bytes in message
SeeAlso: AH=0Ah"Cswitch",AH=0Ch"Cswitch"
--------N-620B--------------------------------
INT 62 - BW-TCP - HW DRIVER (ETHDEV.SYS) - ADD ???
  AH = 0Bh
  AL = ???
  DX = ???
  BP = ???
  ES:SI -> ???
Return: CF clear if successful
  CF set on error
      AL = error code
SeeAlso: AH=0Ch"ETHDEV",AH=FEh,INT 21/AH=3Fh"BW-TCP",INT 63/AH=24h"BW-TCP"
SeeAlso: INT 64/AH=01h"BW-NFS"
--------T-620C--------------------------------
INT 62 - Cswitch - READ MESSAGE
  AH = 0Ch
  CX = number of bytes to read
  DS:SI -> buffer
  DX = queue number (0-63)
Return: AX = status
      FFFFh bad queue number
      else  number of bytes transferred
SeeAlso: AH=0Ah,AH=0Bh
```

```
--------N-620C--------------------------------
INT 62 - BW-TCP - HW DRIVER (ETHDEV.SYS) - REMOVE ???
  AH = 0Ch
  DX = ???
  BP = ???
Return: CF clear if successful
  CF set on error
      AL = error code
Range:  INT 4C to INT FB, selected by configuration
Note: the Beame&Whiteside TCP/IP protocol stack uses two consecutive
    interrupts (62h and 63h by default); the BW-NFS client uses a third
    consecutive interrupt (64h by default) if it is loaded
SeeAlso: AH=0Bh"ETHDEV",AH=FEh,INT 21/AH=3Fh"BW-TCP",INT 63/AH=24h"BW-TCP"
SeeAlso: INT 64/AH=01h"BW-NFS"
--------T-620D--------------------------------
INT 62 - Cswitch - DON'T ALLOW TASK TO BE SWAPPED OUT
  AH = 0Dh
SeeAlso: AH=0Eh"Cswitch"
--------N-620D--------------------------------
INT 62 - BW-TCP - HW DRIVER (ETHDEV.SYS) - NOP for ETHDEV.ODI
  AH = 0Dh
Return: CF clear if successful
  CF set on error
      AL = error code
SeeAlso: AH=FEh,INT 21/AH=3Fh"BW-TCP",INT 63/AH=24h"BW-TCP"
SeeAlso: INT 64/AH=01h"BW-NFS"
--------T-620E--------------------------------
INT 62 - Cswitch - ALLOW TASK TO BE SWAPPED OUT
  AH = 0Eh
SeeAlso: AH=0Dh"Cswitch"
--------N-620E--------------------------------
INT 62 - BW-TCP - HW DRIVER (ETHDEV.SYS) - BEGIN CRITICAL SECTION
  AH = 0Eh
Return: CF clear if successful
  CF set on error
      AL = error code
SeeAlso: AH=0Fh"ETHDEV",AH=10h"ETHDEV",AH=FEh,INT 21/AH=3Fh"BW-TCP"
SeeAlso: INT 63/AH=03h"BW-TCP",INT 63/AH=24h"BW-TCP",INT 64/AH=01h"BW-NFS"
--------T-620F--------------------------------
INT 62 - Cswitch - LOAD AND RUN PROGRAM FROM DISK
  AH = 0Fh
```

```
  ES:BX -> command line
  CX = priority (1-10)
  DX = background flag (nonzero allows loading to EMS)
Return: AX = status
      0000h task loader queue is full
      0001h  no error
SeeAlso: AH=07h"Cswitch",AH=10h"Cswitch",AH=13h"Cswitch"
--------N-620F-------------------------------
INT 62 - BW-TCP - HW DRIVER (ETHDEV.SYS) - END CRITICAL SECTION
  AH = 0Fh
Return: CF clear if successful
  CF set on error
      AL = error code
Range:  INT 4C to INT FB, selected by configuration
Note: the Beame&Whiteside TCP/IP protocol stack uses two consecutive
    interrupts (62h and 63h by default); the BW-NFS client uses a third
    consecutive interrupt (64h by default) if it is loaded
SeeAlso: AH=0Eh"ETHDEV",AH=10h"ETHDEV",AH=FEh,INT 21/AH=3Fh"BW-TCP"
SeeAlso: INT 63/AH=03h"BW-TCP",INT 63/AH=24h"BW-TCP",INT 64/AH=01h"BW-NFS"
--------T-6210-------------------------------
INT 62 - Cswitch - TERMINATE SPAWNED PROGRAM
  AH = 10h
SeeAlso: AH=07h"Cswitch",AH=0Fh"Cswitch"
--------N-6210-------------------------------
INT 62 - BW-TCP - HW DRIVER (ETHDEV.SYS) - QUERY CRITICAL SECTION
  AH = 10h
Return: CF clear if no critical section active
  CF set if in critical section
SeeAlso: AH=0Eh"ETHDEV",AH=0Fh"ETHDEV",AH=FEh,INT 21/AH=3Fh"BW-TCP"
SeeAlso: INT 63/AH=03h"BW-TCP",INT 63/AH=24h"BW-TCP",INT 64/AH=01h"BW-NFS"
--------T-6211-------------------------------
INT 62 - Cswitch - GET TCB INFORMATION
  AH = 11h
  ES:BX -> a pointer which will be set to the tcb address
Return: AX = tcb indentifier
SeeAlso: AH=12h
--------N-6211-------------------------------
INT 62 - BW-TCP - HW DRIVER (ETHDEV.SYS) - SET ???
  AH = 11h
  ES:SI -> ???
Return: CF clear
```

```
SeeAlso: AH=FEh,INT 21/AH=3Fh"BW-TCP",INT 63/AH=24h"BW-TCP"
SeeAlso: INT 64/AH=01h"BW-NFS"
--------T-6212--------------------------------
INT 62 - Cswitch - GET TCB ADDRESS
  AH = 12h
  ES:BX -> a pointer which will be set to the tcb table address
Return: AX = tcb indentifier
SeeAlso: AH=11h"Cswitch"
--------N-6212--------------------------------
INT 62 - BW-TCP - HW DRIVER (ETHDEV.SYS) - GET SOCKET NUMBER???
  AH = 12h
Return: CF clear if successful
      AX = socket number??? (memory variable incremented after reading)
  CF set on error
      AL = error code
Range:  INT 4C to INT FB, selected by configuration
Note: the Beame&Whiteside TCP/IP protocol stack uses two consecutive
    interrupts (62h and 63h by default); the BW-NFS client uses a third
    consecutive interrupt (64h by default) if it is loaded
SeeAlso: AH=FEh,INT 21/AH=3Fh"BW-TCP",INT 63/AH=24h"BW-TCP"
SeeAlso: INT 64/AH=01h"BW-NFS"
--------T-6213--------------------------------
INT 62 - Cswitch - CHECK STATUS OF PREVIOUS LOAD_TASK
  AH = 13h
Return: AX = result
      FFFCh no Memory Control Blocks available
      FFFDh no TCBs available
      FFFEh insufficient memory
      FFFFh cannot open file
      0000h load in progress (not done yet)
      else  tcb indentifier
SeeAlso: AH=0Fh
--------N-6213--------------------------------
INT 62 - BW-TCP - HW DRIVER (ETHDEV.SYS) - ???
  AH = 13h
  CX = ???
Return: CF clear if successful
      AL = 00h
  CF set on error
      AL = error code
SeeAlso: AH=FEh,INT 21/AH=3Fh"BW-TCP",INT 63/AH=24h"BW-TCP"
```

```
SeeAlso: INT 64/AH=01h"BW-NFS"
--------N-6214-------------------------------
INT 62 - BW-TCP - HW DRIVER (ETHDEV.SYS) - ???
  AH = 14h
  ES:SI -> ???
Return: CF clear if successful
      AL = 00h
  CF set on error
      AL = error code
SeeAlso: AH=FEh,INT 21/AH=3Fh"BW-TCP",INT 63/AH=24h"BW-TCP"
SeeAlso: INT 64/AH=01h"BW-NFS"
--------N-6215-------------------------------
INT 62 - BW-TCP - HW DRIVER (ETHDEV.SYS) - GET ???
  AH = 15h
Return: CF clear if successful
      AX = ??? (destroyed???)
  CF set on error
      AL = error code
Range:  INT 4C to INT FB, selected by configuration
Notes:  call this function after reading the "ETHDEV27" device
  the Beame&Whiteside TCP/IP protocol stack uses two consecutive
    interrupts (62h and 63h by default); the BW-NFS client uses a third
    consecutive interrupt (64h by default) if it is loaded
SeeAlso: AH=FEh,INT 21/AH=3Fh"BW-TCP",INT 63/AH=24h"BW-TCP"
INT 64/AH=01h"BW-NFS"
--------N-6216-------------------------------
INT 62 - BW-TCP - HW DRIVER (ETHDEV.SYS) - ???
  AH = 16h
  ???
Return: CF clear if successful
  CF set on error
      AL = error code
SeeAlso: AH=FEh,INT 21/AH=3Fh"BW-TCP",INT 63/AH=24h"BW-TCP"
SeeAlso: INT 64/AH=01h"BW-NFS"
--------N-6217-------------------------------
INT 62 - BW-TCP - HW DRIVER (ETHDEV.SYS) - ???
  AH = 17h
  DX = segment of ???
Return: CF clear
SeeAlso: AH=FEh,INT 21/AH=3Fh"BW-TCP",INT 63/AH=24h"BW-TCP"
SeeAlso: INT 64/AH=01h"BW-NFS"
```

```
--------N-6218--------------------------------
INT 62 - BW-TCP - HW DRIVER (ETHDEV.SYS) - ALLOCATE AND MAP EMS FOR DRIVER
  AH = 18h
Return: CF clear if successful
  CF set on error
      AL = error code
Range:  INT 4C to INT FB, selected by configuration
Notes:  calls function 17h after EMS allocated and mapped
  the Beame&Whiteside TCP/IP protocol stack uses two consecutive
    interrupts (62h and 63h by default); the BW-NFS client uses a third
    consecutive interrupt (64h by default) if it is loaded
SeeAlso: AH=FEh,INT 21/AH=3Fh"BW-TCP",INT 63/AH=03h"BW-TCP",INT 63/AH=24h
SeeAlso: INT 64/AH=01h"BW-NFS"
--------R-6247--------------------------------
INT 62 - PC Tools v7 COMMUTE - ???
  AH = 47h
  AL = subfunction (00h-31h)
  ???
  CF set
Return: ???
Program: COMMUTE is a remote-control program bundled with Central Point
    Software's PC Tools
--------R-6248--------------------------------
INT 62 - PC Tools v7 COMMUTE - ???
  AH = 48h
  AL = ???
  ???
  CF set
Return: ???
--------R-6249--------------------------------
INT 62 - PC Tools v7 COMMUTE - ???
  AH = 49h
  ???
  CF set
Return: ???
Note: may be the same as AH=4Ch
--------R-624A--------------------------------
INT 62 - PC Tools v7 COMMUTE - ???
  AH = 4Ah
  AL = subfunction (00h-46h)
  ???
```

```
  CF set
Return: ???
--------R-624B--BX1234-----------------------
INT 62 - PC Tools v7 COMMUTE - ???
  AH = 4Bh
  BX = 1234h
  CX = 1234h
  ES = ???
  CF set
Return: ???
Program: COMMUTE is a remote-control program bundled with Central Point
    Software's PC Tools
--------R-624C------------------------------
INT 62 - PC Tools v7 COMMUTE - ???
  AH = 4Ch
  BL = subfunction
      00h ???
      02h ???
Return: CF clear if successful
  CF set on error
--------R-626262----------------------------
INT 62 - PC Tools v7 COMMUTE - INSTALLATION CHECK
  AX = 6262h
  CF set
Return: AX = 0000h
  BX = segment of resident code's PSP
Program: COMMUTE is a remote-control program bundled with Central Point
    Software's PC Tools
--------s-62C0------------------------------
INT 62 U - GWBTSR - API
  AH = C0h
  AL = function
      00h installation check
    Return: AX = 00FFh if installed
      01h ???
Program: GWBTSR is a huge (48K) resident mixer controller for the Gateway 2000
    sound card (OEM version of Aztech Sound Galaxy)
Index:  installation check;GWBTSR
--------N-62FE------------------------------
INT 62 - BW-TCP - ETHDRV.SYS - MAP EMS PAGE FRAME
  AH = FEh
```

```
  AL = direction
       00h map in driver's memory block
       01h map out driver's memory block
Return: CF clear if successful
  CF set on error
       AL = error code
Range:  INT 4C to INT FB, selected by configuration
Notes:  this function is supported by at least the SLIP and ODI versions of
    ETHDEV.SYS
  the Beame&Whiteside TCP/IP protocol stack uses two consecutive
    interrupts (62h and 63h by default); the BW-NFS client uses a third
    consecutive interrupt (64h by default) if it is loaded
SeeAlso: INT 21/AH=3Fh"BW-TCP",INT 62/AH=00h"ETHDEV",INT 62/AH=18h"ETHDEV"
SeeAlso: INT 63/AH=03h"BW-TCP",INT 63/AH=24h,INT 64/AH=FEh
--------*-63---------------------------------
INT 63 - reserved for user interrupt
--------d-63---------------------------------
INT 63 - Adaptec and OMTI controllers - DRIVE 0 DATA
Desc: this vector stores the last four bytes of the parameter table for
    hard disk 0
SeeAlso: INT 60"Adaptec",INT 61"Adaptec",INT 62"Adaptec",INT 64"Adaptec"
--------b-63---------------------------------
INT 63 - TI Professional PC - OPTION ROM DATA AREA POINTER (NOT A VECTOR!)
Desc: the low word of this vector contains the segment of the RAM data area
    to be used by the expansion ROM at F400h:4000h, and the high word
    contains the length of the data area; this segment and size are
    both set to 0000h if no ROM is installed at F400h:4000h
SeeAlso: INT 60"TI Professional PC",INT 62"TI Professional"
SeeAlso: INT 64"TI Professional PC"
----------63---------------------------------
INT 63 - Oracle SQL Protected Mode Executive - ???
--------d-63---------------------------------
INT 63 - 4+Power FLOPPY CONTROLLER - ORIGINAL INT 13/40
Desc: the "4+Power" quad floppy controller BIOS hooks INT 13 (or INT 40 if
    INT 13 has been moved there) and places the old value here
----------63---------------------------------
INT 63 - Kofax KF9X00 image manipulation card interface
--------Q-63---------------------------------
INT 63 - DESQview/X - SOCKET API
InstallCheck: test for the string "dvxunix" (yes, lowercase) at offset 9
    from the interrupt handler start (see #03515)
```

```
Notes:  parameters are passed by patching (!) data field immediately following
    the entry point, as detailed below (see #03515); the preferred
    method for calling the socket API is via INT 15/AX=DE2Eh
SeeAlso: INT 15/AX=DE2Eh,INT BE"DESQview"
Index:  installation check;DESQview/X socket interface

Format of DESQview/X socket interrupt handler entry:
Offset  Size  Description (Table 03515)
 00h  3 BYTEs near jump or short jump + NOP to actual interrupt handler
 03h  WORD  offset from following pointer for initial top of local stack
 05h  DWORD pointer to argument/stack block (see INT 15/AX=DE2Eh)
 09h  7 BYTEs signature "dvxunix"
--------b-6300--------------------------------
INT 63 - HP 100LX - MAP HIGH MEMORY
  AH = 00h
  AL = physical page (00h seg C000, 01h seg C400h, ...)
  BX = zero-based logical page
  CX = page number
  DX = device ID (00h system ROM, 05h plugin, etc.)
Return: ???
SeeAlso: AH=01h
--------N-6300--------------------------------
INT 63 - BW-TCP - TCPIP.SYS - SET IP ADDRESS???
  AH = 00h
  DS:BX -> DWORD containing IP address (big-endian)
Return: CF clear if successful
  CF set on error
  AX destroyed
Range:  INT 4D to INT FC, selected by configuration
Note: the Beame&Whiteside TCP/IP protocol stack uses two consecutive
    interrupts (62h and 63h by default); the BW-NFS client uses a third
    consecutive interrupt (64h by default) if it is loaded
SeeAlso: AH=01h"BW-TCP",AH=02h"BW-TCP"
--------b-6301--------------------------------
INT 63 - HP 100LX - SAVE/RESTORE MEMORY MAP
  AH = 01h
  AL = function (00h save, 01h restore)
  ???
Return: ???
--------N-6301--------------------------------
INT 63 - BW-TCP - TCPIP.SYS - ???
```

```
  AH = 01h
  ES:BX -> ???
  ???
Return: ???
Range:  INT 4D to INT FC, selected by configuration
Note: the Beame&Whiteside TCP/IP protocol stack uses two consecutive
    interrupts (62h and 63h by default); the BW-NFS client uses a third
    consecutive interrupt (64h by default) if it is loaded
SeeAlso: AH=00h"BW-TCP",AH=02h"BW-TCP"
--------N-6302--------------------------------
INT 63 - BW-TCP - TCPIP.SYS - ???
  AH = 02h
  ???
Return: ???
SeeAlso: AH=00h"BW-TCP",AH=01h"BW-TCP"
--------N-6303--------------------------------
INT 63 - BW-TCP - TCPIP.SYS - GET IP ADDRESS
  AH = 03h
  DS:SI -> buffer for DWORD IP address (big-endian)
Return: AX destroyed
  CF clear if successful
  CF set on error
Note: this call may use ARP or RARP to determine the address
--------N-6304--------------------------------
INT 63 - BW-TCP - TCPIP.SYS - ???
  AH = 04h
  ???
Return: ???
--------N-6305--------------------------------
INT 63 - BW-TCP - TCPIP.SYS - ???
  AH = 05h
  DS:BX -> ???
  ES:SI -> ???
Return: ???
Range:  INT 4D to INT FC, selected by configuration
--------N-6306--------------------------------
INT 63 - BW-TCP - TCPIP.SYS - ???
  AH = 06h
  ???
Return: ???
--------N-6307--------------------------------
```

```
INT 63 - BW-TCP - TCPIP.SYS - ???
  AH = 07h
  ???
Return: ???
--------N-6308--------------------------------
INT 63 - BW-TCP - TCPIP.SYS - SET DEFAULT ??? HANDLER
  AH = 08h
  DS:BX -> DWORD containing IP address
Return: CF clear if successful
  CF set on error
  ???
--------N-6309--------------------------------
INT 63 - BW-TCP - TCPIP.SYS - INSTALL ??? HANDLERS
  AH = 09h
  BL = handler type
  ES:SI -> FAR handler of specified type
Return: ???
SeeAlso: AH=0Ah,AH=0Dh
--------N-630A--------------------------------
INT 63 - BW-TCP - TCPIP.SYS - DELETE ??? HANDLERS
  AH = 0Ah
  BL = handler type
Return: CF clear if successful
  CF set on error (no handler of specified type installed)
SeeAlso: AH=09h
--------N-630B--------------------------------
INT 63 - BW-TCP - TCPIP.SYS - ???
  AH = 0Bh
  AL = ???
  DL = ???
  DS:BX -> ???
  ES:SI -> ???
Return: ???
--------N-630C--------------------------------
INT 63 - BW-TCP - TCPIP.SYS - ???
  AH = 0Ch
  ???
Return: ???
Range:  INT 4D to INT FC, selected by configuration
--------N-630D--------------------------------
INT 63 - BW-TCP - TCPIP.SYS - INSTALL DEFAULT ??? HANDLER
```

```
  AH = 0Dh
  ???
Return: ???
Note: if not already installed, installs a type 06h handler with AH=09h
SeeAlso: AH=09h
--------N-630E-------------------------------
INT 63 - BW-TCP - TCPIP.SYS - CLOSE NETWORK DESCRIPTOR
  AH = 0Eh
  ???
Return: ???
SeeAlso: INT 61/AH=08h"PC/TCP",INT 61/AH=09h"PC/TCP",INT 61/AH=18h
--------N-630F-------------------------------
INT 63 - BW-TCP - TCPIP.SYS - ???
  AH = 0Fh
  AL = ???
  SI = ???
  DS:DI -> ???
  ???
Return: ???
Range:  INT 4D to INT FC, selected by configuration
--------N-6310-------------------------------
INT 63 - BW-TCP - TCPIP.SYS - ???
  AH = 10h
  DS:DI -> ???
  ???
Return: ???
--------N-6311-------------------------------
INT 63 - BW-TCP - TCPIP.SYS - ???
  AH = 11h
  ???
Return: ???
--------N-6312-------------------------------
INT 63 - BW-TCP - TCPIP.SYS - LISTEN FOR INCOMING CONNECTIONS
  AH = 12h
  DS:SI -> ???
  ES:BP -> ???
Return: ???
SeeAlso: AH=14h,INT 61/AH=23h
--------N-6313-------------------------------
INT 63 - BW-TCP - TCPIP.SYS - NOP
  AH = 13h
```

```
Return: nothing
Range:  INT 4D to INT FC, selected by configuration
--------N-6314--------------------------------
INT 63 - BW-TCP - TCPIP.SYS - OPEN NETWORK CONNECTION
  AH = 14h
  BX = network descriptor???
  DS:SI -> ???
  ES:BP -> ???
Return: ???
SeeAlso: AH=12h,AH=16h,AH=19h,INT 61/AH=13h"PC/TCP",INT 62/AH=13h"ETHDEV"
--------N-6315--------------------------------
INT 63 - BW-TCP - TCPIP.SYS - ???
  AH = 15h
  DS:DI -> ???
  ???
Return: ???
Range:  INT 4D to INT FC, selected by configuration
--------N-6316--------------------------------
INT 63 - BW-TCP - TCPIP.SYS - RESET NETWORK CONNECTION
  AH = 16h
  DS:DI -> ???
Return: ???
Note: calls AH=17h after preprocessing
SeeAlso: AH=17h,INT 61/AH=19h"PC/TCP"
--------N-6317--------------------------------
INT 63 - BW-TCP - TCPIP.SYS - ???
  AH = 17h
  DS:DI -> ???
  ???
Return: ???
Range:  INT 4D to INT FC, selected by configuration
SeeAlso: AH=18h
--------N-6318--------------------------------
INT 63 - BW-TCP - TCPIP.SYS - ???
  AH = 18h
  DS:DI -> ???
  ???
Return: ???
Note: same as AH=17h, except performed with interrupts disabled
SeeAlso: AH=17h
--------N-6319--------------------------------
```

```
INT 63 - BW-TCP - TCPIP.SYS - WRITE TO THE NETWORK
  AH = 19h
  DS:DI -> ???
  ???
Return: BX = number of bytes NOT written
  ???
Range:  INT 4D to INT FC, selected by configuration
Note: calls AH=17h with interrupts disabled and ??? set to 01h
SeeAlso: AH=14h,AH=1Ah,AH=1Bh,INT 61/AH=1Ah"PC/TCP"
--------N-631A--------------------------------
INT 63 - BW-TCP - TCPIP.SYS - READ FROM THE NETWORK
  AH = 1Ah
  CX = maximum number of bytes to read
  ES:BP -> ???
  ???
Return: CX = number of bytes actually read
  ???
SeeAlso: AH=12h,AH=14h,AH=19h,INT 61/AH=1Bh"PC/TCP"
--------N-631B--------------------------------
INT 63 - BW-TCP - TCPIP.SYS - ???
  AH = 1Bh
  CX = ???
  ES:BP -> ???
Return: DX = ???
  ???
Range:  INT 4D to INT FC, selected by configuration
--------N-631C--------------------------------
INT 63 - BW-TCP - TCPIP.SYS - ???
  AH = 1Ch
  DS:DI -> ???
  ???
Return: ???
Note: calls AH=17h with ???
SeeAlso: AH=17h
--------N-631D--------------------------------
INT 63 - BW-TCP - TCPIP.SYS - ???
  AH = 1Dh
  ???
Return: ???
Range:  INT 4D to INT FC, selected by configuration
--------N-631E--------------------------------
```

```
INT 63 - BW-TCP - TCPIP.SYS - ???
  AH = 1Eh
  DS:BX -> DWORD containing IP address (big-endian)
  ???
Return: CF clear if successful
  CF set on error
  ???
--------N-631F--------------------------------
INT 63 - BW-TCP - TCPIP.SYS - SET SOCKET ??? HANDLER
  AH = 1Fh
  BX = socket number
  ES:SI -> FAR function for ???
Return: CF clear if successful
  CF set on error (out of slots)
SeeAlso: AH=20h
--------N-6320--------------------------------
INT 63 - BW-TCP - TCPIP.SYS - REMOVE SOCKET ??? HANDLER
  AH = 20h
  BX = socket number
Return: CF clear if successful
  CF set on error (not set)
Range:  INT 4D to INT FC, selected by configuration
SeeAlso: AH=1Fh
--------N-6321--------------------------------
INT 63 - BW-TCP - TCPIP.SYS - ???
  AH = 21h
  ES:SI -> ???
Return: ???
SeeAlso: INT 61/AH=1Ch"PC/TCP"
--------N-6322--------------------------------
INT 63 - BW-TCP - TCPIP.SYS - REMOVE ??? HANDLER
  AH = 22h
Return: CF clear
Range:  INT 4D to INT FC, selected by configuration
Note: decrements a counter if not already zero, and calls AH=0Ah with BL=11h
    if the counter reaches zero
--------N-6323--------------------------------
INT 63 - BW-TCP - TCPIP.SYS - ???
  AH = 23h
  DS:BX -> ???
  ES:SI -> 6-byte buffer for ???
```

```
Return: CF clear if successful
  CF set on error
--------N-6324--------------------------------
INT 63 - BW-TCP - TCPIP.SYS - GET SOCKET
  AH = 24h
Return: AX = socket number (0400h-FFFFh)
Range:  INT 4D to INT FC, selected by configuration
Note: the Beame&Whiteside TCP/IP protocol stack uses two consecutive
    interrupts (62h and 63h by default); the BW-NFS client uses a third
    consecutive interrupt (64h by default) if it is loaded
SeeAlso: AH=12h,AH=14h,INT 62/AH=12h"ETHDEV",INT 64/AH=01h"BW-NFS"
--------N-6325--------------------------------
INT 63 - BW-TCP - TCPIP.SYS - GET INTERNET ADDRESS
  AH = 25h
Return: CL:CH:DL:DH = caller's Internet address
SeeAlso: AH=03h,AH=26h,INT 61/AH=05h"PC/TCP"
--------N-6326--------------------------------
INT 63 - BW-TCP - TCPIP.SYS - SET INTERNET ADDRESS???
  AH = 26h
  CL:CH:DL:DH = Internet address
Return: nothing
Range:  INT 4D to INT FC, selected by configuration
Note: this function sets a different variable than AH=25h returns
SeeAlso: AH=03h,AH=25h
--------N-6327--------------------------------
INT 63 - BW-TCP - TCPIP.SYS - SET ???
  AH = 27h
  BX = ???
  ES:SI -> ???
Return: ???
--------N-6328--------------------------------
INT 63 - BW-TCP - TCPIP.SYS - ???
  AH = 28h
  ???
Return: ???
--------N-6329--------------------------------
INT 63 - BW-TCP - TCPIP.SYS - ???
  AH = 29h
  ???
Return: ???
Range:  INT 4D to INT FC, selected by configuration
```

```
Note: the Beame&Whiteside TCP/IP protocol stack uses two consecutive
    interrupts (62h and 63h by default); the BW-NFS client uses a third
    consecutive interrupt (64h by default) if it is loaded
--------*-64--------------------------------
INT 64 - reserved for user interrupt
--------d-64--------------------------------
INT 64 - Adaptec controllers - DRIVE 1 DATA
Desc: this vector stores the first four bytes of the parameter table for
    hard disk 1
Notes:  these vectors are used by the following Adaptec controllers:
      ACB 2370 A/B/C, ACB 2372 A/B/C, ACB 2333 A/B, 2322B-8, 2322B-16
  these vectors are NOT used by the following Adaptec controllers:
      ACB 2310, ACB 2312, ACB 2320D, ACB 2322D
SeeAlso: INT 60"Adaptec",INT 65"Adaptec",INT 66"Adaptec",INT 67"Adaptec"
--------b-64--------------------------------
INT 64 - TI Professional PC - OPTION ROM DATA AREA POINTER (NOT A VECTOR!)
Desc: the low word of this vector contains the segment of the RAM data area
    to be used by the expansion ROM at F400h:6000h, and the high word
    contains the length of the data area; this segment and size are
    both set to 0000h if no ROM is installed at F400h:6000h
SeeAlso: INT 60"TI Professional PC",INT 63"TI Professional"
SeeAlso: INT 65"TI Professional PC"
----------64--------------------------------
INT 64 - Oracle SQL Protected Mode Executive - ???
--------N-64--------------------------------
INT 64 - Novell NetWare to v2.0a - LOW-LEVEL API
Note: equivalent to INT 7A for NetWare versions through 2.0a only; later
    versions do not use this interrupt for IPX/SPX access, instead
    getting an entry point from INT 2F/AX=7A00h
SeeAlso: INT 2F/AX=7A00h,INT 7A"LOW-LEVEL API"
--------h-64--------------------------------
INT 64 - Data General DG10 - MicroECLIPSE COPROCESSOR INTERFACE
SeeAlso: INT 65"DG10",INT 66"DG10"
--------r-64--------------------------------
INT 64 - Extended Batch Language v3.14+
  AH = function
       00h to 5Fh chained to previous handler
       60h to 6Ch reserved, return immediately
       80h to FFh chained to previous handler
       6Dh (v4.01+) insert tone in queue
    AL = ???
```

```
CX = frequency in Hertz
DL = duration in clock ticks
Return: AL = 00h if note stored
     = 01h if no room to store
  6Eh clear ??? counter/flag
  6Fh return counter/flag that AH=6Eh clears
  70h ???
AL = ???
  71h ???
AL = ???
  72h ???
  73h insert byte at end of keyboard buffer
AL = byte to insert
Return: AL = 00h if byte inserted
     = 01h if no room to store
  74h insert byte at front of keyboard buffer
AL = byte to insert
Return: AL = 00h if byte inserted
     = 01h if no room to store
  75h ???
  76h get keyboard "stack" status
AL = 'K' if kbd read will read physical keyboard
     'S' if it will read EBL internal keyboard buffer
AH = ???
  77h clear internal keyboard buffer
  78h ???
AL = ???
  79h ???
  7Ah ???
AL = ???
  7Bh ???
AL = ???
  7Ch ???
AL = ???
  7Dh ???
AL = ???
  7Eh clear buffer for ???
  7Fh installation check
Return: CX = version in BCD
  DI = segment of ???
  BX = segment of next program's PSP???
```

```
Program: Extended Batch Language is a batch-file enhancer by Seaware
Notes:  the chaining does not check whether the interrupt had been hooked
    before, so if you try to chain when the previous vector was
    0000h:0000h, you'll be in trouble
  functions 72h and 7Ah-7Dh appear to be interfaces to the optional
    floating-point and extended function packages
Index:  installation check;EBL|installation check;Extended Batch Language
--------d-64---------------------------------
INT 64 - Pdisk by Scott Garfinkle - Overwritten for Hard Drive information
Note: This vector is overwritten by Pdisk to install custom harddrive types.
    It can either destroy 4 vectors and take no memory or TSR and take
    up some memory.
SeeAlso: INT 65"Pdisk"
----------64---------------------------------
INT 64 - PC-DRAFT - ASYNCHRONOUS DRIVER
  ???
Return: ???
Program: PC-DRAFT is a powerful CAD environment by rhv.
SeeAlso: INT 62"PC-DRAFT",INT 65"PC-DRAFT",INT 66"PC-DRAFT",INT 67"PC-DRAFT"
--------N-6401-------------------------------
INT 64 U - BW-NFS - BWRPC - ???
  AH = 01h
  ES:BX -> ??? (at least 8 bytes)
  ES:BP -> DWORD ???
  ???
Return: CF clear if successful
      ???
  CF set on error
      CX = 0000h
Range:  INT 4E to INT FD, selected by configuration
Notes:  the Beame&Whiteside TCP/IP protocol stack uses two consecutive
    interrupts (62h and 63h by default); the BW-NFS client uses a third
    consecutive interrupt (64h by default) if it is loaded
  the BWRPC installation check consists of determining the interrupt
    vector assigned to it (two more than the value returned by reading
    the ETHDEV27 device), and testing whether the word immediately
    preceding the interrupt handler is 4257h ('BW')
SeeAlso: INT 62/AH=00h"ETHDEV",INT 63/AH=03h"BW-TCP",INT 63/AH=24h
Index:  installation checks;BWRPC
--------N-6402-------------------------------
INT 64 U - BW-NFS - BWRPC - ???
```

```
  AH = 02h
  DS:DI -> ???
Return: ???
Note: this call is passed directly through to INT 62/AH=07h
SeeAlso: INT 62/AH=07h"ETHDEV"
--------N-6403------------------------------
INT 64 U - BW-NFS - BWRPC - ADD ???
  AH = 03h
  AL = ???
  BP = ???
  ES:SI -> ???
Return: ???
Note: this call is passed directly through to INT 62/AH=0Bh
SeeAlso: AH=04h,INT 62/AH=0Bh"ETHDEV"
--------N-6404------------------------------
INT 64 U - BW-NFS - BWRPC - REMOVE ???
  AH = 04h
  BP = ???
Return: ???
Range:  INT 4E to INT FD, selected by configuration
Note: this call is passed directly through to INT 62/AH=0Ch
SeeAlso: AH=03h,INT 62/AH=0Ch"ETHDEV"
--------N-6405------------------------------
INT 64 U - BW-NFS - BWRPC - ???
  AH = 05h
  CX = ???
Return: ???
Note: this call is passed directly through to INT 62/AH=13h
SeeAlso: INT 62/AH=13h"ETHDEV"
--------N-6406------------------------------
INT 64 U - BW-NFS - BWRPC - ???
  AH = 06h
  ES:SI -> ???
Return: AL = 00h if CF clear
Range:  INT 4E to INT FD, selected by configuration
Note: this call is passed directly through to INT 62/AH=14h
SeeAlso: INT 62/AH=14h"ETHDEV"
--------N-6407------------------------------
INT 64 U - BW-NFS - BWRPC - GET IP ADDRESS
  AH = 07h
Return: CX:DX = IP address
```

```
--------N-6410--------------------------------
INT 64 U - BW-NFS - BWRPC - CALL ETHDEV.SYS
  AH = 10h
  AL = ETHDEV function number
  other registers as appropriate for ETHDEV call
Return: as returned by ETHDEV
Note: this call is passed directly through to INT 62
SeeAlso: INT 62/AH=00h"ETHDEV"
--------N-6411--------------------------------
INT 64 U - BW-NFS - BWRPC - NOP???
  AH = 11h
Return: CF clear
Range:  INT 4E to INT FD, selected by configuration
--------N-64FE--------------------------------
INT 64 - BW-NFS - BWRPC - MAP EMS PAGE FRAME
  AH = FEh
  AL = direction
       00h map in driver's memory block
       01h map out driver's memory block
Return: CF clear if successful
  CF set on error
       AL = error code
Range:  INT 4E to INT FD, selected by configuration
Note: this call is passed through directly to ETHDEV.SYS (see INT 62/AH=FEh)
SeeAlso: INT 21/AH=3Fh"BW-TCP",INT 62/AH=FEh,INT 63/AH=03h"BW-TCP"
SeeAlso: INT 63/AH=24h
--------*-65----------------------------------
INT 65 - reserved for user interrupt
--------d-65----------------------------------
INT 65 - Adaptec controllers - DRIVE 1 DATA
Desc: this vector stores the second four bytes of the parameter table for
    hard disk 1
SeeAlso: INT 64"Adaptec",INT 66"Adaptec",INT 67"Adaptec"
--------b-65----------------------------------
INT 65 - TI Professional PC - OPTION ROM DATA AREA POINTER (NOT A VECTOR!)
Desc: the low word of this vector contains the segment of the RAM data area
    to be used by the expansion ROM at F400h:8000h, and the high word
    contains the length of the data area; this segment and size are
    both set to 0000h if no ROM is installed at F400h:8000h
SeeAlso: INT 60"TI Professional PC",INT 64"TI Professional"
SeeAlso: INT 66"TI Professional PC"
```

```
--------h-65---------------------------------
INT 65 - Data General DG10 - MicroECLIPSE COPROCESSOR INTERFACE
SeeAlso: INT 64"DG10",INT 66"DG10"
--------N-65---------------------------------
INT 65 - FTP Software NDIS-Packet Driver adapter - POST PROCESSING INTERRUPT
--------U-65---------------------------------
INT 65 - SD.COM v6.2
Desc: The unregistered version of SD62.COM uses the low byte of this vector
    to count the number of invocations, displaying a registration
    reminder each time after the 20th use.
--------d-65---------------------------------
INT 65 - Pdisk by Scott Garfinkle - Overwritten for Hard Drive information
SeeAlso: INT 64"Pdisk",INT 66"Pdisk"
----------65---------------------------------
INT 65 - PC-DRAFT - SECONDARY DISPLAY DRIVER
  ???
Return: ???
Program: PC-DRAFT is a powerful CAD environment by rhv.
SeeAlso: INT 62"PC-DRAFT",INT 64"PC-DRAFT",INT 66"PC-DRAFT",INT 67"PC-DRAFT"
--------s-65---------------------------------
INT 65 - Ad Lib SOUND.COM - INTERFACE
  SI = function number (also see separate entries below)
       0000h Init
       0002h RelTimeStart
       0003h SetState
       0004h GetState
       0005h Flush
       0006h SetMode
       0007h GetMode
       0008h SetRelVolume
       0009h SetTempo
       000Ah SetTranspose
       000Bh GetTranspose
       000Ch SetActVoice
       000Dh GetActVoice
       000Eh PlayNoteDel
       000Fh PlayNote
       0010h SetTimbre
       0011h SetPitch
       0012h SetTickBeat
       0013h NoteOn
```

```
      0014h NoteOff
      0015h Timbre
      0016h SetPitchBend
      0017h WaveForm
  ES:BX -> arguments
InstallCheck: test for the signature block immediately preceding the
    interrupt handler (see #03516)
SeeAlso: SI=8000h
Index:  installation check;Ad Lib SOUND.COM

Format of AdLib signature block:
Offset  Size  Description (Table 03516)
 00h  WORD  version number
 02h 19 BYTEs "SOUND-DRIVER-AD-LIB"
 15h  BYTE  01h
 16h  BYTE  01h
 17h  BYTE  00h
--------N-65----DX4147-----------------------
INT 65 U - NetSoft DOS-NET v1.20+ - ??? API
  DX = 4147h
  BH = 01h
  BL = function number (00h-2Ah)
  ???
Return: CF clear if successful
      AX = ???
  CF set on error
      AX = error code (0001h=invalid function/subfunction)
Range:  INT ?? to INT ??, selected by configuration option; actual interrupt
    number and function number may be retrieved via INT 2A/AX=4147h
Note: this API is supported by CLIENT.COM, SERVER.COM, DOSNET.COM, and
    ROUTER.COM
SeeAlso: DX=4147h"INSTALLATION",INT 2A/AX=4147h
--------N-65----DX4147-----------------------
INT 65 U - NetSoft DOS-NET v1.20+ - ??? API
  DX = 4147h
  BH = 02h
  BL = function number (00h-05h)
  ???
Return: CF clear if successful
      AX = ???
  CF set on error
```

```
      AX = error code (0001h=invalid function/subfunction)
Range:  INT ?? to INT ??, selected by configuration option; actual interrupt
    number and function number may be retrieved via INT 2A/AX=4147h
Note: this API is supported by CLIENT.COM, SERVER.COM, DOSNET.COM, and
    ROUTER.COM
SeeAlso: DX=4147h"INSTALLATION",INT 2A/AX=4147h
--------N-65----DX4147-----------------------
INT 65 U - NetSoft DOS-NET v1.20+ - ??? API
  DX = 4147h
  BH = 03h
  BL = function number (00h-03h) (00h-04h for ROUTER.COM)
  ???
Return: CF clear if successful
      AX = ???
  CF set on error
      AX = error code
Range:  INT ?? to INT ??, selected by configuration option; actual interrupt
    number and function number may be retrieved via INT 2A/AX=4147h
Note: this API is supported by CLIENT.COM, SERVER.COM, DOSNET.COM, and
    ROUTER.COM
SeeAlso: DX=4147h"INSTALLATION",INT 2A/AX=4147h
--------N-65----DX4147-----------------------
INT 65 U - NetSoft DOS-NET v1.20+ - ??? API
  DX = 4147h
  BH = 04h
  BL = function number (00h-07h)
  ???
Return: CF clear if successful
      AX = ???
  CF set on error
      AX = error code
Range:  INT ?? to INT ??, selected by configuration option; actual interrupt
    number and function number may be retrieved via INT 2A/AX=4147h
Note: this API is supported by CLIENT.COM, SERVER.COM, and DOSNET.COM
SeeAlso: DX=4147h"INSTALLATION",INT 2A/AX=4147h
--------N-65----DX4147-----------------------
INT 65 U - NetSoft DOS-NET v1.20+ - ??? API
  DX = 4147h
  BH = 07h
  BL = function number (00h-08h)
  ???
```

```
Return: CF clear if successful
      AX = ???
  CF set on error
      AX = error code (0001h=invalid function/subfunction)
Range:  INT ?? to INT ??, selected by configuration option; actual interrupt
    number and function number may be retrieved via INT 2A/AX=4147h
Note: this API is supported by CLIENT.COM, SERVER.COM, and DOSNET.COM
SeeAlso: DX=4147h"INSTALLATION",INT 2A/AX=4147h
--------N-65----DX4147-----------------------
INT 65 U - NetSoft DOS-NET v1.20+ - INSTALLATION CHECK???
  DX = 4147h
  BH = 80h
Return: CF set
  AX = 0001h
Range:  INT ?? to INT ??, selected by configuration option; actual interrupt
    number and function number may be retrieved via INT 2A/AX=4147h
Note: this call is supported by CLIENT.COM, SERVER.COM, and ROUTER.COM
SeeAlso: DX=4147h,INT 2A/AX=4147h
--------N-65----DX4741-----------------------
INT 65 U - NetSoft DOS-NET v1.20+ - SPOOLER - ???
  DX = 4741h
  BH = 01h
  AL = 02h
  BL = function number (0Eh,0Fh)
  AH = subfunction number
  ???
Return: ???
Range:  INT ?? to INT ??, selected by configuration option; actual interrupt
    number may be retrieved via INT 2A/AX=4147h
SeeAlso: DX=4147h"INSTALLATION",INT 2A/AX=4147h
--------N-65----DX4741-----------------------
INT 65 U - NetSoft DOS-NET v1.20+ - PRNREDIR - ???
  DX = 4741h
  BH = 01h
  AL = 03h
  BL = function number (0Eh,0Fh)
  AH = subfunction number
  ???
Return: ???
Range:  INT ?? to INT ??, selected by configuration option; actual interrupt
    number may be retrieved via INT 2A/AX=4147h
```

```
SeeAlso: DX=4147h"INSTALLATION",INT 2A/AX=4147h
--------N-65----DX4741-----------------------
INT 65 U - NetSoft DOS-NET v1.20+ - PRNREDIR - ???
  DX = 4741h
  BH = 01h
  AL = 04h
  BL = function number (0Eh,0Fh)
  AH = subfunction number
  ???
Return: ???
Range:  INT ?? to INT ??, selected by configuration option; actual interrupt
    number may be retrieved via INT 2A/AX=4147h
SeeAlso: DX=4147h"INSTALLATION",INT 2A/AX=4147h
--------N-65----DX4741-----------------------
INT 65 U - NetSoft DOS-NET v1.20+ - NETBIOS - ???
  DX = 4741h
  BH = 01h
  AL = 07h
  BL = function number (0Eh,0Fh)
  AH = subfunction number
  ???
Return: ???
Range:  INT ?? to INT ??, selected by configuration option; actual interrupt
    number may be retrieved via INT 2A/AX=4147h
SeeAlso: DX=4147h"INSTALLATION",INT 2A/AX=4147h
--------N-65----DX4741-----------------------
INT 65 U - NetSoft DOS-NET v1.20+ - MACTEST - ???
  DX = 4741h
  BH = 01h
  AL = 08h
  BL = function number (0Eh,0Fh)
  AH = subfunction number
  ???
Return: ???
Range:  INT ?? to INT ??, selected by configuration option; actual interrupt
    number may be retrieved via INT 2A/AX=4147h
SeeAlso: DX=4147h"INSTALLATION",INT 2A/AX=4147h
--------N-65----DX4741-----------------------
INT 65 U - NetSoft DOS-NET v1.20+ - Physical Layer - ???
  DX = 4741h
  BH = 02h
```

```
  BL = function number (01h-04h)
  ???
Return: ???
  ---function 02h---
  DS:SI -> ??? data area
Range:  INT ?? to INT ??, selected by configuration option; actual interrupt
    number may be retrieved via INT 2A/AX=4147h
Note: this API is supported by PARALLEL.COM, SERIAL.COM, ARCNET.COM,
    ETHERNET.COM, NDIS.COM, ODI.COM, SMC.COM, and FTP.COM
SeeAlso: DX=4147h"INSTALLATION",INT 2A/AX=4147h
--------N-65----DX4741-----------------------
INT 65 U - NetSoft DOS-NET v1.20+ - SPOOLER.COM - ???
  DX = 4741h
  BH = 05h
  BL = function number (00h,01h)
Return: CF clear
  AL = status code (00h=successful)
Range:  INT ?? to INT ??, selected by configuration option; actual interrupt
    number may be retrieved via INT 2A/AX=4147h
SeeAlso: DX=4147h"INSTALLATION",INT 2A/AX=4147h
--------N-65----DX4741-----------------------
INT 65 U - NetSoft DOS-NET v1.20+ - PRNREDIR.COM - ???
  DX = 4741h
  BH = 06h
  BL = function number (00h-04h)
  ???
Return: ???
Range:  INT ?? to INT ??, selected by configuration option; actual interrupt
    number may be retrieved via INT 2A/AX=4147h
SeeAlso: DX=4147h"INSTALLATION",INT 2A/AX=4147h
--------N-65----DX4741-----------------------
INT 65 U - NetSoft DOS-NET v1.20+ - COMREDIR.COM - ???
  DX = 4741h
  BH = 08h
  BL = function number (00h-01h)
  ???
Return: ???
Range:  INT ?? to INT ??, selected by configuration option; actual interrupt
    number may be retrieved via INT 2A/AX=4147h
SeeAlso: DX=4147h"INSTALLATION",INT 2A/AX=4147h
--------N-65----DX4741-----------------------
```

```
INT 65 U - NetSoft DOS-NET v1.20+ - FTP.COM - ???
  DX = 4741h
  BX = 8010h
  AL = instance number???
Return: AX = 0008h if AL matches internal variable (call chained otherwise)
Range:  INT ?? to INT ??, selected by configuration option; actual interrupt
    number may be retrieved via INT 2A/AX=4147h
SeeAlso: DX=4147h"INSTALLATION",INT 2A/AX=4147h
--------s-65----SI0000----------------------
INT 65 - Ad Lib SOUND.COM - INITIALIZE (RESET)
  SI = 0000h
--------s-65----SI0003----------------------
INT 65 - Ad Lib SOUND.COM - SET STATE
  SI = 0003h
  ES:BX -> WORD new state (0000h disabled, 0001h enabled)
SeeAlso: SI=0004h
--------s-65----SI0004----------------------
INT 65 - Ad Lib SOUND.COM - GET STATE
  SI = 0004h
Return: AX = status
      0000h all done playing sounds
      else  still playing sounds
SeeAlso: SI=0003h
--------s-65----SI0006----------------------
INT 65 - Ad Lib SOUND.COM - SET MODE
  SI = 0006h
  ES:BX -> WORD new mode (0000h melodic, 0001h percussive)
SeeAlso: SI=0007h
--------s-65----SI0007----------------------
INT 65 - Ad Lib SOUND.COM - GET MODE
  SI = 0007h
Return: AX = mode
      0000h melodic
      0001h percussive
SeeAlso: SI=0006h
--------s-65----SI000C----------------------
INT 65 - Ad Lib SOUND.COM - SET ACTIVE VOICE
  SI = 000Ch
  ES:BX -> WORD voice = 0000h to 0008h
SeeAlso: SI=000Dh
--------s-65----SI000D----------------------
```

```
INT 65 - Ad Lib SOUND.COM - GET ACTIVE VOICE
  SI = 000Dh
Return: AX = voice (0000h to 0008h)
SeeAlso: SI=000Ch
--------s-65----SI8000-----------------------
INT 65 u - Media Vision FM.COM v4.1a+ - GET INTERNAL DATA STRUCTURES
  SI = 8000h
Return: DX:AX -> internal data structures
Program: FM.COM is an Ad Lib SOUND.COM-compatible driver for Media Vision's
    Pro Audio Spectrum sound boards
SeeAlso: SI=8001h
--------s-65----SI8001-----------------------
INT 65 u - Media Vision FM.COM v4.1a+ - GET VOICE COUNT
  SI = 8001h
Return: AX = ???
  DX = number of voices??? (09h or 0Bh)
SeeAlso: SI=8000h
--------s-65----SI8002-----------------------
INT 65 - Media Vision FM.COM v4.1a+ - START BACKGROUND FM SOUNDS
  SI = 8002h
SeeAlso: SI=8003h
--------s-65----SI8003-----------------------
INT 65 - Media Vision FM.COM v4.1a+ - STOP BACKGROUND FM SOUNDS
  SI = 8003h
SeeAlso: SI=8002h
--------s-65----SI8004-----------------------
INT 65 U - Media Vision FM.COM v4.1a+ - GET ???
  SI = 8004h
Return: AX = ??? (0280h)
  DX = ??? (01A0h)
--------s-65----SI8005-----------------------
INT 65 U - Media Vision FM.COM v4.1a+ - ???
  SI = 8005h
  ???
Return: ???
SeeAlso: SI=8000h
--------S-65---------------------------------
INT 65 U - EZRECV v1.0 - API
  AX = function
      0000h ???
    Return: AX = ??? or FFFFh
```

```
      0001h ???
    Return: AX = status (0000h or 0001h)
      0002h ???
    Return: AX = status (0000h or 0001h)
      0003h set ??? to 0001h
    Return: AX = 0000h
      0004h ???
    Return: AX = ???
Return: BH = COM port being used
  BL = speed???
  CH = ???
  CL = ???
  DX = ???
  DS = ???
  ES = EZRECV data segment
Program: EZRECV is a background Zmodem file receiver by Express Consulting
--------*-66---------------------------------
INT 66 - reserved for user interrupt
--------d-66---------------------------------
INT 66 - Adaptec controllers - DRIVE 1 DATA
Desc: this vector stores the third four bytes of the parameter table for
    hard disk 1
SeeAlso: INT 64"Adaptec",INT 65"Adaptec",INT 67"Adaptec"
--------b-66---------------------------------
INT 66 - TI Professional PC - SYSTEM INFORMATION (NOT A VECTOR!)
Desc: the low word of this vector contains the system memory size in
    paragraphs; the third byte contains the number of outstanding
    interrupt requests, and the fourth byte contains a description
    of the installed drive types (see #03517)
SeeAlso: INT 60"TI Professional PC",INT 67"TI Professional"

Bitfields for TI Professional drive type information:
Bit(s)  Description (Table 03517)
 7  floppy drive D: has 80 tracks
 6  floppy drive D: is double-sided
 5  floppy drive C: has 80 tracks
 4  floppy drive C: is double-sided
 3  floppy drive B: has 80 tracks
 2  floppy drive B: is double-sided
 1  floppy drive A: has 80 tracks
 0  floppy drive A: is double-sided
```

```
--------h-66---------------------------------
INT 66 - Data General DG10 - MicroECLIPSE COPROCESSOR INTERFACE
SeeAlso: INT 64"DG10"
--------N-66---------------------------------
INT 66 C - Nanosoft, Inc. TurboNET - NETWORK PROCESSING ???
Program: TurboNET is a NetBIOS-based file redirector and server
Note: hooked but not used (IRET) by both redirector and server; called from
    server's INT 28 handler
SeeAlso: INT 2F/AX=8100h
--------d-66---------------------------------
INT 66 - Pdisk by Scott Garfinkle - Overwritten for Hard Drive information
SeeAlso: INT 64"Pdisk",INT 67"Pdisk"
--------W-66---------------------------------
INT 66 - Microsoft Windows VITD.386 Virtual Interval Timer
Note: This Windows 3.x Virtual Device Driver implements a virtual timer
    which will expire and call INT 66.  This timer can be used to
    calculate elapsed execution time etc.
--------K-66---------------------------------
INT 66 - Newkey v5.4 - INSTALLATION VECTOR
Return: immediately (IRET)
Program: Newkey is a shareware keyboard macro program by Frank A. Bell
InstallCheck: test for the signature bytes FDh FCh FFh FEh at offset 03h in
    the interrupt handlers segment
Range:  INT 60h to INT 67h, selected by scanning for highest unused vector
BUG:  the code obviously intends to use INT F0-FE, INT 70-77, and INT 68-6F
    before falling back to INT 60-67, but only uses the last of these
    ranges in v5.4
SeeAlso: INT 2F/AX=E300h
Index: installation checks;Newkey|Newkey;installation check
----------66---------------------------------
INT 66 - PC-DRAFT - TABLET/DIGITIZER DRIVER
  ???
Return: ???
Program: PC-DRAFT is a powerful CAD environment by rhv.
SeeAlso: INT 62"PC-DRAFT",INT 64"PC-DRAFT",INT 65"PC-DRAFT",INT 67"PC-DRAFT"
--------U-66---------------------------------
INT 66 - PC-Magazin - INCA
  details not yet availble
Program: INCA is a utility from PC-Magazin (the German edition of PC Magazine)
    issue 51-52/85.
SeeAlso: INT 61"SWAPx"
```

```
--------F-6601--------------------------------
INT 66 - BitFax Scheduler - SET MODE???
  AH = 01h
SeeAlso: AH=02h
--------F-6602--------------------------------
INT 66 - BitFax Scheduler - SET MODE???
  AH = 02h
SeeAlso: AH=01h
--------F-6603--------------------------------
INT 66 - BitFax Scheduler - SCHEDULE FAX TRANSMISSIONS
  AH = 03h
  ???
Return: ???
SeeAlso: AH=05h
--------F-6604--------------------------------
INT 66 - BitFax Scheduler - GET STATUS???
  AH = 04h
Return: AX = ??? (0000h or 0001h)
  DX = BitSched version???  (for versions >= 3.00)
       9796h (ver. 3.00)
       97E6h (ver. 3.02)
       92D0h (ver. 3.04.06)
       9510h (ver. 3.06.02)
SeeAlso: AH=06h,AX=3345h,INT 2F/AX=8000h"FaxBIOS"
--------F-6605--------------------------------
INT 66 - BitFax Scheduler - CONVERT FILE AND SEND FAX
  AH = 05h
  BX:CX -> command block (see #03518)
  ???
Return: ???
SeeAlso: AH=03h

Format of BitFax command block:
Offset  Size  Description (Table 03518)
 00h 18 BYTEs configuration bytes???
 12h  BYTEs ASCIZ temporary file name to place converted fax
 52h  BYTEs ASCIZ directory containing BitFax executables
 92h  BYTEs ASCIZ telephone number
 C2h  BYTE  cover page control (00h don't send, 01h do send cover page)
 C3h 15 BYTEs configuration bytes???
 E2h  BYTEs ASCIZ path of BITFAX.TRA file (containing additional
```

```
      configuration information???)
122h  BYTEs configuration bytes???
12Ch  BYTE  00h don't send cover page
    01h send cover page
12Dh  7 BYTEs configuration bytes???
134h  BYTEs ASCIZ path of file to send
174h  BYTEs more configuration bytes???
  ???
--------F-6606--------------------------------
INT 66 - BitFax Scheduler - SET MODE???
  AH = 06h
Return: DX = BitSched version??? (same as AH=04h)
SeeAlso: AH=04h
--------s-660688------------------------------
INT 66 - IBMSND driver, DIGPAK - PLAY 8-BIT DIGITIZED SOUND
  AX = 0688h
  DS:SI -> SNDSTRUC (see #03519)
Return: ???
Program: The IBMSND driver is part of John W. Ratcliff's
     The IBM Digitized Sound Package
  DIGPAK is a set of digitized sound drivers written by
    John W. Ratcliff, The Audio Solution, Inc.
InstallCheck: test for a valid signature string six bytes prior to the
    interrupt handler; this string may be either "KERN" or "MIDI" (in
    the latter case, call AX=0701h to determine whether IBMSND is
    installed)
SeeAlso: AX=068Bh,AX=068Fh,AX=0701h

Format of IBMSND driver SNDSTRUC:
Offset  Size  Description (Table 03519)
 00h  DWORD -> audio data
 04h  WORD  length of audio data in bytes
 06h  DWORD -> playback status flag
 0Ah  WORD  playback frequency
--------s-660689------------------------------
INT 66 - IBMSND driver, DIGPAK - REPORT SOUND DRIVER STATUS
  AX = 0689h
Return: AX = status
      0000h no sound playing
      0001h sound effect is currently playing
  ---DIGPAK---
```

```
  BX = version number (v3.1+)
  DX = looping status
       0000h no sound looping
       0001h sound effect is currently looping
SeeAlso: AX=0688h,AX=068Bh,AX=068Ch
Index:  version check;DIGPAK
--------s-66068A-----------------------------
INT 66 - IBMSND driver, DIGPAK - PREFORMAT SOUND
  AX = 068Ah
  DS:SI -> SNDSTRUC (see #03519)
Desc: convert audio data into output hardware format
SeeAlso: AX=068Bh
--------s-66068B-----------------------------
INT 66 - IBMSND driver, DIGPAK - PLAY PREFORMATTED SOUND
  AX = 068Bh
  DS:SI -> SNDSTRUC (see #03519)
Return: AX = ???
SeeAlso: AX=0688h,AX=068Ah,AX=068Fh
--------s-66068C-----------------------------
INT 66 - IBMSND driver, DIGPAK - REPORT AUDIO DRIVER CAPABILITIES
  AX = 068Ch
Return: AX = capabilities (see #03520)
  DX = playback rate if fixed-frequency playback
  ---DIGPAK---
  BX:CX -> ASCIZ ID string
SeeAlso: AX=0689h,AX=068Dh

Bitfields for IBMSND driver capabilities:
Bit(s)  Description (Table 03520)
 0  can play audio in background
 1  data is massaged for output device
 2  driver plays at fixed frequency, resampling input data to fit
 3  driver uses timer interrupt
---DIGPAK---
 4  device supports timer sharing
 5  supports looped sounds and pending
 6  supports stereo panning
 7  supports 8-bit PCM stereo playback
 8  supports audio recording
 9  supports DMA bakcfilling
--------s-66068D-----------------------------
```

```
INT 66 - IBMSND driver, DIGPAK - REPORT CURRENT SAMPLE ADDRESS
  AX = 068Dh
Return: AX = current playback address
Desc: determine what point in the audio data the playback has reached, for
    synchronization with video or animation effects
Notes:  this function applies to background playback only
  the reported address may be an approximation rather than the exact
    address
SeeAlso: AX=068Ch,AX=0691h
--------s-66068E-----------------------------
INT 66 - IBMSND driver, DIGPAK - SET CALLBACK ADDRESS
  AX = 068Eh
  BX:DX -> callback function
      0000h:0000h to disable callback
  DS = value to load into DS when calling the callback function
Desc: specify the function to be called when playback of a sound effect is
    completed
Note: the callback function will typically be called during a hardware
    interrupt, so all the usual precautions should be taken except for
    preserving registers
SeeAlso: AX=0691h
--------s-66068F-----------------------------
INT 66 - IBMSND driver, DIGPAK - STOP CURRENT SOUND
  AX = 068Fh
Desc: cause any currently-playing sound effect to be terminated
SeeAlso: AX=0688h,AX=068Bh
--------s-660690-----------------------------
INT 66 - IBMSND driver, DIGPAK - "SetAudioHardware" - SET UP HARDWARE INFO
  AX = 0690h
  BX = IRQ
  CX = base address
  DX = other setup value (device-dependent???)
--------s-660691-----------------------------
INT 66 - IBMSND driver, DIGPAK - REPORT CALLBACK ADDRESS
  AX = 0691h
Return: AX:DX -> current callback function
  BX = original caller's DS register
Program: The IBMSND driver is part of John W. Ratcliff's
     The IBM Digitized Sound Package
  DIGPAK is a set of digitized sound drivers written by
    John W. Ratcliff, The Audio Solution, Inc.
```

```
SeeAlso: AX=068Eh
--------s-660693-----------------------------
INT 66 - DIGPAK - SET TIMER DIVISOR RATE
  AX = 0693h
  DX = rate
Program: DIGPAK is a set of digitized sound drivers written by
    John W. Ratcliff, The Audio Solution, Inc.
--------s-660694-----------------------------
INT 66 - DIGPAK - PLAY PREFORMATTED DATA
  AX = 0694h
  DS:SI -> Sound Data structure (see #03519)
Return: AX = status???
--------s-660695-----------------------------
INT 66 - DIGPAK - POST AUDIO PENDING
  AX = 0695h
  DS:SI -> Sound Data structure (***)
Return: AX = status
      0000h sound started playing
      0001h sound was posted as pending to play
      0002h sound effect already pending, this one not posted
SeeAlso: AX=0696h
--------s-660696-----------------------------
INT 66 - DIGPAK - GET AUDIO PENDING STATUS
  AX = 0696h
Return: AX = status
      0000h no sound is playing
      0001h sound playing, and a sound is pending
      0002h sound playing, no sound pending
SeeAlso: AX=0695h
--------s-660697-----------------------------
INT 66 - DIGPAK - SET STEREO PANNING
  AX = 0697h
  DX = panning position (0 = right, 127 = left)
Return: AX = status
      0000h command ignored (not supported)
      0001h panning set
--------s-660698-----------------------------
INT 66 - DIGPAK - SET PLAY MODE
  AX = 0698h
  DX = playback mode
      0000h 8-bit PCM
```

```
      0001h 8-bit stereo PCM
      0002h 16-bit PCM
      0003h 16-bit stereo PCM
Return: AX = status
      0000h command ignored
      0001h mode set
--------s-660699-----------------------------
INT 66 - DIGPAK - GET ADDRESSES
  AX = 0699h
Return: AX = pending address
  BX = semaphore address
--------s-66069A-----------------------------
INT 66 - DIGPAK - SET RECORD MODE
  AX = 069Ah
  DX = recording mode
      0000h turn audio recording on
      0001h turn audio recording off
Return: AX = status
      0000h command ignored
      0001h audio recording mode set
--------s-66069B-----------------------------
INT 66 - DIGPAK - STOP NEXT LOOP
  AX = 069Bh
--------s-66069C-----------------------------
INT 66 - DIGPAK - SET DMA BACKFILL MODE
  AX = 069Ch
  DX = mode
      0000h turn backfill mode on
      0001h turn backfill mode off
Return: AX = status
      0000h command ignored
      0001h backfill mode set
SeeAlso: AX=069Dh,AX=069Eh
--------s-66069D-----------------------------
INT 66 - DIGPAK - REPORT DMA COUNTER
  AX = 069Dh
Return: AX = DMA counter
SeeAlso: AX=069Eh
--------s-66069E-----------------------------
INT 66 - DIGPAK - VERIFY DMA BLOCK
  AX = 069Eh
```

```
  CX = length of buffer
  ES:BX -> buffer containing sound data
Return: AX = status
      0000h block crosses 64K bounadary
      0001h block is OK
SeeAlso: AX=069Dh
--------s-66069F-----------------------------
INT 66 - DIGPAK - SET PCM VOLUME
  AX = 069Fh
  BX = left channel volume (0-100)
  CX = right channel volume (0-100)
Return: AX =  status
      0000h command ignored
      0001h volume set
--------s-6606A0-----------------------------
INT 66 - DIGPAK - SET DPMI MODE
  AX = 06A0h
  DX = mode
      0000h 32-bit register addressing on
      0001h 32-bit register addressing off
SeeAlso: INT 31/AX=0400h
--------s-660700-----------------------------
INT 66 - MIDPAK - UNINSTALL
  AX = 0700h
Note: this function should NOT be called by applications
Program: MIDPAK is a set of MIDI sound drivers developed by Miles Design
    Incorporated.
--------s-660701-----------------------------
INT 66 - IBM Digitized Sound Package MIDI driver - GET DIGITIZED SOUND CAPABIL
  AX = 0701h
Return: AX = digitized sound capabilities
      0000h if digitized sound driver (functions 06xxh) not available
InstallCheck: test for the signature "MIDI" six bytes before the interrupt
    handler
Note: also supported by MIDPAK, the successor to the Digitized Sound
    Package's MIDI driver
SeeAlso: AX=0688h
--------s-660702-----------------------------
INT 66 - MIDPAK - PLAY SEQUENCE
  AX = 0702h
  BX = Sequence number
```

```
Return: AX = status
      0000h Sequence is being played
      0001h Sequence not available
SeeAlso: AX=0703h,AX=0705h
--------s-660703-----------------------------
INT 66 - MIDPAK - SEGUE SEQUENCE
  AX = 0703h
  BX = sequence number
  CX = activation code (FFFFh is next trigger)
Return: ???
--------s-660704-----------------------------
INT 66 - MIDPAK - REGISTER XMIDI
  AX = 0704h
  CX:BX -> XMIDI sequence data
  DI:SI = length of XMIDI data
Return: AX = status
      0000h unable to register XMIDI data
      0001h XMIDI file registered resident
      0002h XMIDI file was registered to the application
--------s-660705-----------------------------
INT 66 - MIDPAK - STOP MIDI
  AX = 0705h
SeeAlso: AX=0702h,AX=0709h
--------s-660706-----------------------------
INT 66 O - MIDPAK - REMAP CHANNEL
  AX = 0706h
  BX = sequence
  CX = physical
--------s-660707-----------------------------
INT 66 - MIDPAK - REPORT TRIGGER EVENT COUNTER
  AX = 0707h
Return: AX = count of number of callbacks since last reset
  DX = ID
SeeAlso: AX=0708h,AX=0713h
--------s-660708-----------------------------
INT 66 - MIDPAK - RESET EVENT TRIGGER COUNTER
  AX = 0708h
SeeAlso: AX=0707h,AX=0713h
--------s-660709-----------------------------
INT 66 O - MIDPAK - MIDI SLEEP
  AX = 0709h
```

```
SeeAlso: AX=070Ah
--------s-66070A-----------------------------
INT 66 O - MIDPAK - MIDI AWAKE
  AX = 070Ah
SeeAlso: AX=0709h
--------s-66070B-----------------------------
INT 66 - MIDPAK - RESUME PLAYING
  AX = 070Bh
SeeAlso: AX=070Ch
--------s-66070C-----------------------------
INT 66 - MIDPAK - GET SEQUENCE STATUS
  AX = 070Ch
Return: AX = status
      0000h sequence stopped
      0001h sequence playing
      0002h sequence done
SeeAlso: AX=070Bh
--------s-66070D-----------------------------
INT 66 - MIDPAK - REGISTER XMIDI FILE
  AX = 070Dh
  CX:BX -> ASCII filename
SeeAlso: AX=0704h,AX=0710h
--------s-66070E-----------------------------
INT 66 - MIDPAK - GET RELATIVE VOLUME
  AX = 070Eh
Return: AX = current volume
SeeAlso: AX=070Fh
--------s-66070F-----------------------------
INT 66 - MIDPAK - SET RELATIVE VOLUME
  AX = 070Fh
  BX = new volume
  CX = time
SeeAlso: AX=070Eh
--------s-660710-----------------------------
INT 66 - MIDPAK - LOAD MIDPAK DRIVER
  AX = 0710h
  BX = segment of .ADV driver
  CX = 0000h (offset must be zero)
  DX:SI -> .AD driver
SeeAlso: AX=070Dh
--------s-660711-----------------------------
```

```
INT 66 - MIDPAK - POLL MIDPAK
  AX = 0711h
Return: AX = ???
  ???
SeeAlso: AX=0712h
--------s-660712----------------------------
INT 66 - MIDPAK - GET MIDI CLOCK
  AX = 0712h
Return: AX:DX = clock counter
  CX:BX = clock address
SeeAlso: AX=0711h,AX=0713h
--------s-660713----------------------------
INT 66 - MIDPAK - GET TRIGGER COUNT ADDRESS
  AX = 0713h
Return: AX:DX -> trigger counter address
SeeAlso: AX=0707h,AX=0712h,AX=0714h
--------s-660714----------------------------
INT 66 - MIDPAK - GET EVENT ID ADDRESS
  AX = 0714h
Return: AX:DX -> event ID
SeeAlso: AX=0713h,AX=0716h
--------s-660716----------------------------
INT 66 - MIDPAK - REPORT SEQUENCE NUMBER
  AX = 0716h
Return: AX = current sequence number
SeeAlso: AX=0702h
Program: MIDPAK is a set of MIDI sound drivers developed by Miles Design
    Incorporated.
--------n-6610----------------------------
INT 66 - PenDOS - TDMOUSE.EXE - GET ???
  AH = 10h
Return: CF clear
  AX = 0000h
  BX = ??? (0012h)
  DX:CX -> TDMOUSE INT 33 handler (IRET to hide mouse from other apps)
Program: TDMOUSE is a PenDOS hardware driver which allows a mouse to emulate
    a touchpad; PenDOS is a set of programs by Communication Intelligence
    Corporation which makes applications pen-aware
--------n-6611----------------------------
INT 66 - PenDOS - TDMOUSE.EXE - SET ??? HANDLER
  AH = 11h
```

```
  DX:BX -> new handler for ???
Return: CF clear
  AX = 0000h
  DX:BX -> old handler for ??? (points at RETF by default)
--------n-6612--------------------------------
INT 66 - PenDOS - TDMOUSE.EXE - INITIALIZE
  AH = 12h
Return: CF clear
  AX = 0000h
Note: this function calls the old mouse handler with functions 0000h, 0002h,
    0007h, 0008h, 000Fh, 0004h, and 000Ch (in that order)
SeeAlso: AH=13h
--------n-6613--------------------------------
INT 66 - PenDOS - TDMOUSE.EXE - SHUTDOWN???
  AH = 13h
Return: CF clear
  other register as returned by INT 33/AX=0000h
SeeAlso: AH=12h
--------n-6614--------------------------------
INT 66 - PenDOS - TDMOUSE.EXE - ???
  AH = 14h
  BX = ???
  CX = ???
Return: CF clear
  AX = 0000h
--------n-6615--------------------------------
INT 66 - PenDOS - TDMOUSE.EXE - SET ??? HANDLER
  AH = 15h
  DX:BX -> new handler for ???
Return: CF clear
  AX = 0000h
  DX:BX -> old handler (points at RETF by default)
--------n-6616--------------------------------
INT 66 - PenDOS - TDMOUSE.EXE - UNUSED FUNCTIONS
  AH = 16h to 1Fh
Return: CF set
Program: TDMOUSE is a PenDOS hardware driver which allows a mouse to emulate
    a touchpad; PenDOS is a set of programs by Communication Intelligence
    Corporation which makes applications pen-aware
--------n-6621--------------------------------
INT 66 - PenDOS - PINK - ???
```

```
  AH = 21h
Return: CF clear if successful
  CF set on error
Note: this function sets ??? flag or counter (also set by AH=2Fh) to FFFFh
--------n-6622--------------------------------
INT 66 - PenDOS - PINK - ???
  AH = 22h
  DX:BX -> ???
  CL = ???
Return: CF clear if successful
  CF set on error
  ???
SeeAlso: AH=24h
--------n-6623--------------------------------
INT 66 - PenDOS - PINK - ???
  AH = 23h
  ???
Return: CF clear if successful
  CF set on error
  ???
--------n-6624--------------------------------
INT 66 - PenDOS - PINK - ???
  AH = 24h
  DX:BX -> ???
  CL = ???
Return: CF clear if successful
  CF set on error
  ???
SeeAlso: AH=22h
--------n-6625--------------------------------
INT 66 - PenDOS - PINK - ???
  AH = 25h
  CL = ??? (NOP if 00h)
  ???
Return: CF clear if successful
  CF set on error
  ???
--------n-6627--------------------------------
INT 66 - PenDOS - PINK - ???
  AH = 27h
  BL = ???
```

```
  BH = ???
  CL = ??? (0-3)
  DL = ??? (> BL)
  DH = ??? (> BH)
Return: ???
--------n-6628-------------------------------
INT 66 - PenDOS - PINK - ???
  AH = 28h
  ???
Return: CF clear if successful
  CF set on error
  ???
Note: this function sets ??? flag or counter (also set by AH=2Fh) to FFFFh
--------n-6629-------------------------------
INT 66 - PenDOS - PINK - ???
  AH = 29h
  ???
Return: ???
Note: this function sets ??? flag or counter (also set by AH=2Fh) to FFFFh
--------n-662A-------------------------------
INT 66 - PenDOS - PINK - ???
  AH = 2Ah
  DL = ??? (nonzero)
  DH = ??? (nonzero)
Return: CF clear if successful
  CF set on error
  ???
--------n-662B-------------------------------
INT 66 - PenDOS - PINK - ???
  AH = 2Bh
  ???
Return: CF clear if successful
  CF set on error
  ???
--------n-662F-------------------------------
INT 66 - PenDOS - PINK - INITIALIZE
  AH = 2Fh
  ???
Return: AX = status
      0000h failed
      FFFFh successful
```

```
  ???
Note: this function sets ??? flag or counter to FFFFh and hooks INT 1Ch
--------F-663345-----------------------------
INT 66 - BitFax Scheduler - REMOVE TSR FROM MEMORY
  AX = 3345h
Return: AX = FFFFh error removing TSR
InstallCheck: test for the signature "BitFax Scheduler" beginning two bytes
    past the interrupt handler
SeeAlso: AH=04h,INT 2F/AH=2Ah,INT 2F/AX=CB00h
Index:  installation check;BitFax Scheduler
--------n-6640-------------------------------
INT 66 - PenDOS - PKEYUS - GET VERSION
  AH = 40h
Return: CF clear
  AX = 0000h
  BH = major version (02h for version bundled with IBM DOS 6.1)
  BL = minor version (00h for version bundled with IBM DOS 6.1)
  DL = ??? (4Eh)
  DH = ??? (0Eh)
--------n-6641-------------------------------
INT 66 - PenDOS - PKEYUS - SET ???
  AH = 41h
  BX = ???
  CL = ??? (08h-20h)
  DL = screen column??? (<= 50h)
  DH = screen row???  (<= 3Ch)
Return: AX = status (0000h successful, 0001h error)
Note: this function also sets an internal flag
SeeAlso: AH=42h,AH=43h
--------n-6642-------------------------------
INT 66 - PenDOS - PKEYUS - ???
  AH = 42h
Return: CF clear
  AX = 0000h
Note: this function also clears the flag set by AH=41h
SeeAlso: AH=41h
--------n-6643-------------------------------
INT 66 - PenDOS - PKEYUS - ???
  AH = 43h
  BX = ???
  DX = ???
```

```
Return: AX = status
      0000h if AH=41h flag set
      else
    AH = ???
    AL = ???
    BX = ???
    DX = ???
SeeAlso: AH=41h
--------n-6644--------------------------------
INT 66 - PenDOS - PKEYUS - UNUSED FUNCTIONS
  AH = 44h to 4Fh
Return: CF set
--------n-6650--------------------------------
INT 66 - PenDOS - PMOUSE - SET ???
  AH = 50h
  BX = ???
  CH = ???
  DX = ???
Return: CF clear
  AX = 0000h
--------n-6651--------------------------------
INT 66 - PenDOS - PMOUSE - NOP
  AH = 51h
Return: CF set
--------n-6652--------------------------------
INT 66 - PenDOS - PMOUSE - ???
  AH = 52h
  BX = ???
  CL = ???
  DX = ???
Return: ???
--------n-6653--------------------------------
INT 66 - PenDOS - PMOUSE - UNUSED FUNCTIONS
  AH = 53h to 57h
Return: CF set
--------n-66----------------------------------
INT 66 - PenDOS - PMOUSE - ALTERNATE API
  AH = function (58h-5Fh)
Note: these functions exactly duplicate AH=50h-57h
--------U-66AA02------------------------------
INT 66 - HelpTSR v2.10 - INSTALLATION CHECK
```

```
  AX = AA02h
Return: ES:DI -> 7 byte signature "HelpTSR" if resident
Program: HelpTSR is a resident viewer by David Jurgens for HelpPC
--------n-66C5-------------------------------
INT 66 - PenDOS - VLOAD - API
  AH = C5h
  ???
Return: ???
--------t-66FFFBBXFFFB-----------------------
INT 66 - MicroHelp Stay-Res Plus - ???
  AX = FFFBh
  BX = FFFBh
  ???
Return: ???
SeeAlso: AX=FFFEh,INT 2D"AMIS"
--------t-66FFFEBXFFFE-----------------------
INT 66 - MicroHelp Stay-Res/Stay-Res Plus - UNINSTALL
  AX = FFFEh
  BX = FFFEh
Return: only if unsuccessful
InstallCheck: test whether the interrupt handler begins with the bytes
    FBh 9Ch or 9Ch FAh, and the program name (not case-sensitive) appears
    at offset 0005h (older versions) or the offset returned by
    AX=FFFFh/BX=FFF0h in the interrupt handler segment
Note: Programs which use Stay-Res include ThesPlus (program name "THESPLUS")
    and Personal Calendar (program name "CAL") by Paul Mun~oz-Colman.
SeeAlso: AX=FFFBh,AX=FFFFh,INT 2D"AMIS"
Index:  installation check;MicroHelp Stay-Res|installation check;ThesPlus
Index:  installation check;Personal Calendar|installation check;CAL
--------t-66FFFFBXFFF0-----------------------
INT 66 - MicroHelp Stay-Res Plus - FIND PROGRAM NAME
  AX = FFFFh
  BX = FFF0h
Return: DI = offset of program name in interrupt handler segment
SeeAlso: AX=FFFBh,AX=FFFEh,INT 2D"AMIS"
--------d-67---------------------------------
INT 67 - Adaptec controllers - DRIVE 1 DATA
Desc: this vector stores the last four bytes of the parameter table for
    hard disk 1
SeeAlso: INT 64"Adaptec",INT 65"Adaptec",INT 66"Adaptec"
--------b-67---------------------------------
```

```
INT 67 - TI Professional PC - SYSTEM DATA (NOT A VECTOR!)
Desc: this vector contains the TI Pro's system configuration words
    (see #03521)
SeeAlso: INT 66"TI Professional PC"

Bitfields for TI Professional PC System Configuration doubleword:
Bit(s)  Description (Table 03521)
 0  8087 present
 31-1 reserved (0)
--------d-67---------------------------------
INT 67 - Pdisk by Scott Garfinkle - Overwritten for Hard Drive information
SeeAlso: INT 64"Pdisk",INT 66"Pdisk"
--------I-67---------------------------------
INT 67 - Sangoma CCPOP 3270 resident module
SeeAlso: INT 61"Sangoma",INT 68"Sangoma"
--------U-67---------------------------------
INT 67 - CUCKOO.COM - INSTALLATION CHECK
Program: CUCKOO is a resident on-screen clock with optional hourly chime or
    cuckoo by an unknown author with revisions by Thomas A. Lundin
Note: this is not a vector; when loaded for the first time, CUCKOO.COM uses
    the last unused (0000h:0000h) vector in the range 60h-67h to store
    the signature value 434Ch:4F4Bh ('CLOK')
----------67---------------------------------
INT 67 - PC-DRAFT - KEYBOARD DRIVER
  ???
Return: ???
Program: PC-DRAFT is a powerful CAD environment by rhv.
SeeAlso: INT 62"PC-DRAFT",INT 64"PC-DRAFT",INT 65"PC-DRAFT",INT 66"PC-DRAFT"
--------N-6700-------------------------------
INT 67 - PC-NET, Alloy NTNX - LOCK SEMAPHORE AND WAIT
  AH = 00h
  DS:DX -> ASCIZ semaphore name (max 64 bytes)
Return: AL = status (see #03522)
  AH = semaphore owner if status=02h
SeeAlso: AH=01h,AH=02h"PC-NET",INT 7F/AH=00h

(Table 03522)
Values for PC-NET semaphore function status:
 00h  successful
 01h  invalid function
 02h  semaphore already locked
```

```
 03h  unable to lock semaphore
 04h  semaphore space exhausted
--------N-6701-------------------------------
INT 67 - PC-NET, Alloy NTNX - LOCK SEMAPHORE
  AH = 01h
  DS:DX -> ASCIZ semaphore name (max 64 bytes)
Return: AL = status (see #03522)
  AH = semaphore owner if status=02h
SeeAlso: AH=00h,AH=02h"PC-NET",INT 7F/AH=01h"Alloy"
--------N-6702-------------------------------
INT 67 - PC-NET, Alloy NTNX - UNLOCK SEMAPHORE
  AH = 02h
  DS:DX -> ASCIZ semaphore name (max 64 bytes)
Return: AL = status (see #03522)
  AH = semaphore owner if status=02h
SeeAlso: AH=00h,AH=01h"PC-NET",INT 7F/AH=02h
--------m-671E-------------------------------
INT 67 U - Qualitas 386MAX v7.00 - MEMLIMIT - INSTALLATION CHECK
  AH = 1Eh
Return: AH = 00h if installed
      AL destroyed
      ES:DI -> ASCII signature "MemLimit"
SeeAlso: AH=1Fh,INT 21/AX=4402h"386MAX"
--------m-671F-------------------------------
INT 67 U - Qualitas 386MAX v7.00 - MEMLIMIT - API
  AH = 1Fh
  DS:SI -> request packet (see #03523)
Return: AH = status (00h successful, 84h invalid function code, etc.)
SeeAlso: AH=1Eh

Format of 386MAX MEMLIMIT request packet:
Offset  Size  Description (Table 03523)
 00h  WORD  function code (00h-0Fh)
 02h  WORD  return code (see #03524)
 04h  4 BYTEs ???
 08h  WORD  ???
  ???

(Table 03524)
Values for 386MAX MEMLIMIT return code:
 00h  unknown request
```

```
 01h  invalid parameter for VCPI limit
 02h  VCPI limit set
 03h  invalid parameter for EMS limit
 04h  EMS limit set
 05h  DPMI disabled
 06h  XMS disabled
 07h  XMS limit set
 08h  unable to uninstall
 09h  unloaded
--------m-672763CL01-------------------------
INT 67 - VIDEMS.SYS v1.31+ - INSTALLATION CHECK
  AX = 2763h
  CL = 01h
  BX = signature AAFFh
Return: AH = 00h if VIDEMS is installed
      CH = 00h if optimization for 80286 is used, 01h otherwise
      CL = internal revision number (typically 00h to 03h)
      DX = driver version (DH=major, DL=minor; not a BCD!)
  AH = 84h if not installed but EMS manager is present
Program: VIDEMS is an expanded memory manager from Conea Software Corp. It
   converts video adapter RAM to LIM 3.2 EMS.
SeeAlso: AX=2763h/CL=02h,AX=2763h/CL=03h
--------m-672763CL02-------------------------
INT 67 - VIDEMS.SYS v1.31+ - FLUSH EMS TO VIDEO RAM
  AX = 2763h
  CL = 02h
  BX = signature AAFFh
Return: AH = 00h if successful
Notes:  This call is normally used by Conea products only.
SeeAlso: AX=2763h/CL=01h,AX=2763h/CL=03h
--------m-672763CL03-------------------------
INT 67 - VIDEMS.SYS v1.31+ - RELOAD EMS FROM VIDEO RAM
  AX = 2763h
  CL = 03h
  BX = signature AAFFh
Return: AH = 00h if successful
Notes:  This call is normally used by Conea products only.
SeeAlso: AX=2763h/CL=01h,AX=2763h/CL=03h
--------m-672763CL04-------------------------
INT 67 - VIDEMS.SYS v1.31+ - RETURN HIDDEN BLOCK SIZE
  AX = 2763h
```

```
  CL = 04h
  BX = signature AAFFh
Return: AH = 00h if successful
      DX = block size in kilobytes
Desc: Returns the amount of EMS which can be safely used at any time, and
    can't be destroyed by writing anything to the B800:0000 buffer.
Note: this function normally returns 184K, while the driver provides up to
    240K of EMS.
SeeAlso: AX=2763h/CL=01h,AX=2763h/CL=02h
--------m-672763CL05-------------------------
INT 67 - VIDEMS.SYS v1.31+ - RETURN PHYSICAL PAGE SIZE
  AX = 2763h
  CL = 05h
  BX = signature AAFFh
Return: AH = 00h
  DX = size in bytes
Desc: Used by Conea products to determine if addressing mode has changed.
    Normally, all VIDEMS versions beginning from 1.15 use 4K "physical"
    pages.
Notes:  this function will probably become raw page size in future releases of
    VIDEMS, which will support the LIM 4.0 standard.
SeeAlso: AX=2763h/CL=01h,AX=2763h/CL=02h
--------m-672763CL06-------------------------
INT 67 - VIDEMS.SYS v1.50+ - RETURN VIDEO RAM SIZE
  AX = 2763h
  CL = 06h
  BX = signature AAFFh
Return: AH = 00h
  DX = total amount of video RAM
Desc: beginning with version 1.50, VIDEMS can use more adapter memory on
    some chipsets, and is no longer limited to the first 240K.
Notes:  this call is not officially documented, and Conea programmers use it
    mainly for debugging purposes.
--------u-672833-----------------------------
INT 67 - Q87 v4+ - PREPARE TO UNLOAD AND GET XMS HANDLE FOR Q87 MEMORY
  AX = 2833h
  EAX = 29482833h (entire EAX value required)
Return: EAX = XMS handle for memory allocated when Q87 was installed
  EBX = status
      00000000h Q87 is in demo mode (countdown running);
        Q87 remains active
```

```
      00000001h Q87 is in registered mode; memory manager's IDT, GDT,
        and optionally CR0 (if BL <> 5Fh on entry) have
        been restored
      00000002h Q87 is in demo mode (demo time has expired);
        Q87 remains active
Note: this call is used by UNLOAD87 to release the memory used by Q87
BUG:  v4.00-v4.03 will hang on most machines when run under bare DOS with
    no memory manager, because neither Q87 nor UNLOAD87 checks whether
    there is a valid INT 67 handler before performing an installation
    check via INT 67
SeeAlso: AX=4321h,INT 21/AX=4321h/BX=0000h
--------m-673F--CX5145-----------------------
INT 67 U - QEMM-386 v4.23+ - INSTALLATION CHECK
  AH = 3Fh
  CX = 5145h ("QE")
  DX = 4D4Dh ("MM")
Return: AH = 00h if installed
      ES:DI -> QEMM API entry point (see #03525,#03528,#03640)
Notes:  if no other program has hooked INT 67, an alternate installation
    check is to test for the string
    "QUARTERDECK EXPANDED MEMORY MANAGER 386" at offset 14h in the INT 67
    handler's segment; the word at offset 12h contains the offset in
    the handler's segment of the API entry point
  although this function is still undocumented, Quarterdeck has recently
    documented two alternate methods for determining the QEMM API entry
    point, as well as several of the API functions
  MICEMM (Micronics Expanded Memory Manager) versions 2.0C and 4D support
    the alternate QEMM installation check and entry point functions 00h,
    02h, and 03h; version 4D only provides the signature string if the
    commandline argument "DV" is provided
  386MAX v6.01 responds to this call, but DESQview 2.42 does not
    recognize the returned entry point as providing QEMM's capabilities
    because a) only functions 0Ch (different from QEMM 0Ch) and
      1000h-1009h are supported,
      b) status is returned as for EMS functions, not QEMM funcs
      c) the protected-mode entry point returned by function 1000h
      only supports functions 0Ch, 1004h, 1005h, and 100Ah
  the string check mentioned above is not supported by 386MAX
SeeAlso: AX=5BF0h,AH=DDh,AX=FFA5h,INT 15/AX=11DEh,INT 21/AX=4402h/SF=01h
SeeAlso: INT 21/AX=4402h"QEMM",INT 21/AX=4402h"386MAX",INT 2F/AX=D201h/BX=5145h
```

```
(Table 03525)
Values for calling QEMM "QPI_GetStatus" function:
  AH = 00h get QEMM state
Return: CF clear
  AL = QEMM state
       bit 0 set if QEMM turned OFF
       bit 1 set if in "Auto" mode
Note: this function is officially documented
SeeAlso: #03526,#03527,#03528,#03640

(Table 03526)
Values for calling QEMM "QPI_SetStatus" function:
  AH = 01h set QEMM state
  AL = new state
       bit 0 set: place QEMM in OFF state
Return: CF clear if successful
  CF set on error
Note: this function is officially documented
SeeAlso: #03525

(Table 03527)
Values for calling QEMM QPI function 02h:
  AH = 02h get ???
Return: CF clear
  AX = segment of ??? data structure
  Data Structure
  Offset  Size  Description
   00h  DWORD page table entry for ???
    ???
SeeAlso: #03528

(Table 03528)
Values for calling QEMM "QPI_GetVersion" function:
  AH = 03h get QEMM version
Return: CF clear
  AX = BX = version in BCD
Notes:  this function is officially documented.  The most recent official docs
    state that the version is returned in both AX and BX; older
    documentation only mentions BX
  MICEMM returns AX=0001h, BX unchanged
SeeAlso: #03525,#03529
```

```
(Table 03529)
Values for calling QEMM QPI function 04h:
  AH = 04h allocate 4K page and set AUTO/ON mode
Return: CF clear if successful
      DX = page number of a 4K page
  CF set if unable to allocate page
Note: QEMM mode unchanged if not AUTO/OFF
SeeAlso: #03530

(Table 03530)
Values for calling QEMM QPI function 05h:
  AH = 05h free 4K page and turn QEMM off
  DX = page number returned by function 04h
Return: CF clear
Note: QEMM mode unchanged if not AUTO/ON
SeeAlso: #03529,#03531

(Table 03531)
Values for calling QEMM QPI function 06h:
  AH = 06h make new mapping context???
  DX = page number of 4K page to hold page table
Return: CF clear
Note: copies page table into given page and then sets ??? page table entry
    to point at copy
SeeAlso: #03532,#03533

(Table 03532)
Values for calling QEMM QPI function 07h:
  AH = 07h get mapping context
Return: CF clear
  DX = page number of page table for current mapping context
SeeAlso: #03528,#03531,#03533

(Table 03533)
Values for calling QEMM QPI function 08h:
  AH = 08h set mapping context???
  DX = linear page number of page table
Return: CF clear
SeeAlso: #03528,#03531,#03532,#03534,#03536
```

```
(Table 03534)
Values for calling QEMM QPI function 09h:
  AH = 09h get linear page number for page table entry
  CX = page table index
Return: CF clear
  DX = linear page number
SeeAlso: #03535

(Table 03535)
Values for calling QEMM QPI function 0Ah:
  AH = 0Ah set linear page number for page table entry
  CX = page table index
  DX = linear page number
Return: CF clear
SeeAlso: #03534

(Table 03536)
Values for calling QEMM QPI function 0Bh:
  AH = 0Bh map 4K pages
  BX = number of pages
  CX = first page number (must be 0100h to allocate HMA)
  DX = EMS handle (memory belonging to EMS handle will be mapped
        into the address space beginning with the first page
        allocated to the handle)
Return: AH = 00h
SeeAlso: #03533,#03537

(Table 03537)
Values for calling QEMM QPI function 0Ch:
  AH = 0Ch get available memory
Return: CF clear
  BX = 0001h
  CX = total 4K pages???
  DX = number of 4K pages free
SeeAlso: #03536,#03538

(Table 03538)
Values for calling QEMM QPI function 0Dh:
  AH = 0Dh CRT controller I/O port trapping
  AL = mode
       00h only trap on I/O ports 03C0h-03C5h, 03C7h, 03CAh-03CFh
```

```
      01h trap on ports 03B4h, 03B5h, 03B8h, 03C6h, 03C8h, 03C9h,
      03D4h, and 03D5h
      02h only trap on I/O ports 03C6h, 03C8h, and 03C9h
Return: CF clear

(Table 03539)
Values for calling QEMM QPI function 0Eh:
  AH = 0Eh set cursor virtualization callbacks
  DS:BX -> FAR routine for getting hardware cursor address
  ES:DX -> FAR routine for setting hardware cursor address
Return: CF clear
Note: both callbacks are invoked with CL indicating which
    CRT controller register to access (0Eh for high
    byte of cursor address, 0Fh for low byte)
  the DS:BX callback should return BX=cursor address;
    ES:DX is called with BL or BH (depending on CL)
    set to the appropriate half of the cursor's address

(Table 03540)
Values for calling QEMM QPI function 0Fh:
  AH = 0Fh unmap 4K pages
  CX = first page number
  DX = number of pages
Return: CF clear
  AL = 00h/01h if ???
Note: if CX=0100h and DX=0010h, the HMA is remapped to
    simulate a disabled A20

(Table 03541)
Values for calling QEMM QPI function 1000h:
  AX = 1000h get protected-mode interface
  DS:SI -> 16-byte buffer for two GDT entries
  ES:DI -> buffer for 4K page table
Return: CF clear
  EAX = offset of protected-mode API entry point
  DS:SI buffer filled with two GDT descriptors
    first is QEMM code segment, second is data???
  ES:DI buffer filled with 4K page table
  DI points to first unused page table entry
SeeAlso: INT 67/AX=DE01h
```

```
(Table 03542)
Values for calling QEMM QPI function 1001h:
  AX = 1001h get CPU debug registers
  ES:DI -> buffer for debug registers (8 DWORDs)
Return: CF clear
  BL = INT01 handling (see #03543)
  ES:DI buffer filled

(Table 03543)
Values for calling QEMM QPI function 1002h:
  AX = 1002h set CPU debug registers
  BL = INT01 handling
       00h  reflect all debugging exceptions as V86-mode INT 01's
       else convert debugging exceptions other than single-step
       into V86-mode INT 03's, single-step to INT 01's
  ES:DI -> buffer containing debug registers (8 DWORDs)
Return: CF clear
Notes:  identical to INT 67/AX=DE09h if BL=01h
  the INT01 handling flag is set to 01h by the general-protection
    violation handler for certain privileged instructions
SeeAlso: #03542

(Table 03544)
Values for calling QEMM QPI function 1003h:
  AX = 1003h get machine status word CR0
Return: CF clear
  EAX = contents of CR0
SeeAlso: INT 67/AX=DE07h

(Table 03545)
Values for calling QEMM QPI function 1004h:
  AX = 1004h allocate a 4K page
Return: CF clear if successful
       EDX = linear address of allocated page
  CF set on error
SeeAlso: INT 67/AX=DE04h

(Table 03546)
Values for calling QEMM QPI function 1005h:
  AX = 1005h free 4K page
  EDX = linear address of page to free
```

```
Return: CF clear
SeeAlso: INT 67/AX=DE05h

(Table 03547)
Values for calling QEMM QPI function 1006h:
  AX = 1006h NOP
Return: CF set

(Table 03548)
Values for calling QEMM QPI function 1007h:
  AX = 1007h get maximum physical memory address
Return: CF clear
  EDX = physical address of highest 4K memory page
SeeAlso: INT 67/AX=DE02h

(Table 03549)
Values for calling QEMM QPI function 1008h:
  AX = 1008h get physical address of page in first megabyte
  CX = page number (linear address shifted right 12 bits)
Return: CF clear
  EDX = linear address of page
SeeAlso: #03548,#03624,#03625

(Table 03550)
Values for calling QEMM QPI function 1009h:
  AX = 1009h switch to protected mode
  ESI = linear address in first megabyte of system reg values
        (see INT 67/AX=DE0Ch)
  interrupts disabled
Return: interrupts disabled
  GDTR, IDTR, LDTR, TR loaded
  SS:ESP must have at least 16 bytes space, and the
        entry point is required to set up a new stack
        before enabling interrupts
  EAX, ESI, DS, ES, FS, GS destroyed

(Table 03551)
Values for calling QEMM QPI function 100Ah:
  AX = 100Ah switch back to virtual-86 mode
  DS = selector for data segment from function 1000h
  SS:ESP in first megabyte of linear memory
```

```
  interrupts disabled
  STACK:  QWORD  return address from FAR call to 32-bit segment
    DWORD  EIP
    DWORD  CS
    DWORD  reserved for EFLAGS
    DWORD  ESP
    DWORD  SS
    DWORD  ES
    DWORD  DS
    DWORD  FS
    DWORD  GS
  will switch to virtual86 mode with interrupts disabled, all
    segment registers loaded, and EAX destroyed.

(Table 03552)
Values for calling QEMM QPI function 11h:
  AH = 11h get memory type map
  AL = zero/nonzero ??? (set by QEMM.COM but apparently ignored
       by QEMM 6.00)
  ES:DI -> 256-byte buffer for memory types
Return: CF clear
  BL = ???
  ES:DI buffer filled
Note: each byte of the buffer corresponds to a 4K page, and
    contains the type of that page: 00h = mappable,
    02h = mapped ROM, 03h = high RAM, 04h = excluded,
    05h = video, 06h = ROM, 07h = adapter ROM,
    08h = split ROM, 09h = page frame, 0Ah = RAMmable,
    0Bh = conventional, 83h = high RAM under MS Windows

(Table 03553)
Values for calling QEMM QPI function 12h:
  AH = 12h get HIRAM chain
Return: CF clear
  BX = segment of first MCB in high memory
      0000h if no high memory

(Table 03554)
Values for calling QEMM QPI function 1300h:
  AX = 1300h VIDRAMEGA???
  BL = 00h copy ???
```

```
       nonzero copy ??? (reverse)
Return: CF clear
  AL = status
       00h if all pages clean
       01h if any page dirty

(Table 03555)
Values for calling QEMM QPI function 1301h:
  AX = 1301h check if pages modified
  DX:DI = start address of range to check
  CX = length of range in paragraphs
Return: CF clear
  CX = status
       0000h none of the indicated pages is dirty
    DI destroyed
       1000h one or more pages is dirty
    DI = low word of first dirty page's linear addr

(Table 03556)
Values for calling QEMM QPI function 1302h:
  AX = 1302h ???
  BL = ???
  BH = ???
  CX = ???
  SI = offset of ???
  DI = offset of ???
  ???
Return: CF clear
  ???
Note: disables certain interrupts at the two 8259 PICs during
    execution; also modifies CRT controller during
    execution under certain circumstances

(Table 03557)
Values for calling QEMM QPI function 1303h:
  AX = 1303h initialize EGA graphics virtualization
  BX = number of pages (less 1) of EMS to allocate
Return: CF clear if successful
       DX = EMS handle
  CF set on error
```

```
(Table 03558)
Values for calling QEMM QPI function 1304h:
  AX = 1304h shutdown EGA graphics virtualization
  DX = EMS handle being used for virtualization
Return: CF clear

(Table 03559)
Values for calling QEMM QPI function 1305h:
  AX = 1305h select portion of EGA graphics to virtualize???
  (related to graphics virtualization, changes memory mappings)
  CX = start offset within A000h segment of virtualized mem???
Return: CF clear
Note: disables certain interrupts at the two 8259 PICs during
    execution (see #03566) and runs inside a QEMM
    critical section

(Table 03560)
Values for calling QEMM QPI function 1306h:
  AX = 1306h set DESQview critical section counter address
  ES:BX -> WORD DESQview critical section counter or 0000h:0000h
Return: CF clear
Note: also sets a pointer in the low-memory part of QEMM to
    the current value of INT 15 if ES:BX not 0000h:0000h

(Table 03561)
Values for calling QEMM QPI function 1307h:
  AX = 1307h ??? (changes memory mappings for entire A000h segment)
Return: CF clear
Note: disables certain interrupts at the two 8259 PICs during
    execution (see #03566) and runs inside a QEMM
    critical section

(Table 03562)
Values for calling QEMM QPI function 1308h:
  AX = 1308h start/reset CRT controller I/O trapping
  BL = subfunction
      00h restore CRTC I/O port trapping to previous state
      else start trapping all accesses to I/O ports 03B0-03DF
Return: CF clear
Note: if called more than once in a row with BL nonzero, the
    original state of the I/O port trapping will be lost
```

```
(Table 03563)
Values for calling QEMM QPI function 1309h:
  AX = 1309h Hercules mode-change support
  ES:BX -> new address for Hercules mode-change callback
Return: CF clear
Note: the callback function is called whenever the CRTC mode
    register is written, with AL set to the value written

(Table 03564)
Values for calling QEMM QPI function 130Ah:
  AX = 130Ah virtualize EGA/VGA DAC registers (I/O ports 03C8h/03C9h)
  CX:DX -> DAC register virtualization buffer (see #03647)
    or 0000h:0000h to disable
Return: CF clear

(Table 03565)
Values for calling QEMM QPI function 130Bh:
  AX = 130Bh ???
  BL = ??? (??? or 00h)
Return: CF clear
  ???
Note: calls AX=130Eh in some cases

(Table 03566)
Values for calling QEMM QPI function 130Ch:
  AX = 130Ch set interrupts to mask
  BX = interrupts to mask out during AX=1302h,AX=1307h,AX=1308h,
       AX=130Dh,AX=1310h (BL = master PIC, BH = slave PIC)
Return: CF clear

(Table 03567)
Values for calling QEMM QPI function 130Dh:
  AX = 130Dh map EGA memory at A0000h
  ???
Return: CF clear
Note: disables certain interrupts at the two 8259 PICs during execution
    (see #03566) and runs inside a QEMM critical section
  calls AX=1307h (see #03561)

(Table 03568)
```

```
Values for calling QEMM QPI function 130Eh:
  AX = 130Eh ??? (modifies CRT controller setup)
  ???
Return: CF clear

(Table 03569)
Values for calling QEMM QPI function 130Fh:
  AX = 130Fh reset ???
Return: CF clear

(Table 03570)
Values for calling QEMM QPI function 1310h:
  AX = 1310h copy modified pages to physical video RAM???
  ???
Return: CF clear
Note: disables certain interrupts at the two 8259 PICs during execution
    (see #03566) and runs inside a QEMM critical section
  also calls AX=130Dh (see #03567)

(Table 03571)
Values for calling QEMM QPI function 1311h:
  AX = 1311h set ???
  BL = zero/nonzero???
Return: CF clear
Note: certain operations will be performed with interrupts
    (as set by AX=130Ch) enabled rather than disabled if
    called with BL nonzero

(Table 03572)
Values for calling QEMM QPI function 1312h:
  AX = 1312h (v6.02) NOP???
Note: called by DV 2.42, but appears to be a NOP in QEMM 6.02

(Table 03573)
Values for calling QEMM QPI function 1400h:
  AX = 1400h initialize DESQview "protection level" support
  ES:DI -> protection level configuration (at least 24 bytes)
    (see #03641)
  BL = highest ??? to return (one less than number of words)
Return: CF clear
  AX = ??? (4204h for v6.00)
```

```
Note: QEMM also sets the protected mode INT 02 and INT 06
    vectors to alternate handlers in certain cases

(Table 03574)
Values for calling QEMM QPI function 1401h:
  AX = 1401h turn off DESQview protection level support
    Return: CF clear
      ???
    Notes:  clears the DV critical-section flag address set with
        function 1306h
      QEMM also sets the protected mode INT 02 and INT 06
        vectors to the default handlers if they had been
        revectored by function 1400h

(Table 03575)
Values for calling QEMM QPI function 1402h:
  AX = 1402h set protection level???
    BL = protection level???
        00h NOP
        01h ???
        02h ???
        other (03h) ???
    ES:DI -> ???
    Return: CF clear
      ???
    Format of Data structure:
    Offset  Size  Description
     00h  WORD  segment of ??? (X, word at X:0136h set to X)
     02h  WORD  segment of ??? (word at X:0124h set to this)
     04h  WORD  number of paragraphs of ???
     06h  3 WORDs ??? (copied to X:0000h)
     0Ch  WORD  ???

(Table 03576)
Values for calling QEMM QPI function 1403h:
  AX = 1403h add ??? to end of list and ??? (execute func 1406h)
    ES:DI -> ??? structure added to end of ??? list
      (at least 31 bytes, DWORD at offset 06h used for
       storing pointer to next struc, WORD at offset 00h
       seems to be a key or index)
    Return: CF clear
```

```
(Table 03577)
Values for calling QEMM QPI function 1404h:
  AX = 1404h NOP
    Return: CF clear

(Table 03578)
Values for calling QEMM QPI function 1405h:
  AX = 1405h remove ??? from ??? list
    BX = key???
    Return: CF clear

(Table 03579)
Values for calling QEMM QPI function 1406h:
  AX = 1406h ???
    ???
    Return: CF clear
      ???
    Notes:  this function is a NOP unless protection level 2 or 3
        is active
      when not a NOP, one of the actions is to write-protect
        certain memory pages

(Table 03580)
Values for calling QEMM QPI function 1407h:
  AX = 1407h ???
    ???
    Return: CF clear
      ???
    Note: same as function 1406h, but only does anything if
        protection level 2 is active

(Table 03581)
Values for calling QEMM QPI function 1408h:
  AX = 1408h unprotect???
    ???
    Return: CF clear
      ???

(Table 03582)
Values for calling QEMM QPI function 1409h:
```

```
  AX = 1409h abort program causing protection violation???
    ???
    Return: CF clear
      ???

(Table 03583)
Values for calling QEMM QPI function 140Ah:
  AX = 140Ah set ???
    BX = index of ???
    Return: CF clear
      ???
    Notes:  no range checking is performed on BX
      this function is a NOP unless protection level 3 active

(Table 03584)
Values for calling QEMM QPI function 140Bh:
  AX = 140Bh get ???
    BX = index of ???
    SI = 0000h
    Return: CF clear
      SI = segment of 256-byte buffer??? or 0000h
    Notes:  no range checking is performed on BX
      this function is a NOP unless protection level 3 active

(Table 03585)
Values for calling QEMM QPI function 15h:
  AH = 15h set timer channel 0 virtualization buffer
    ES:BX -> WORD buffer for timer channel 0 divisor
      0000h:0000h to disable virtualization
    Return: CF clear

(Table 03586)
Values for calling QEMM v5.00+ QPI function 1600h:
  AX = 1600h get memory access status
    ES:DI -> 256-byte buffer
    Return: ES:DI buffer filled
    Note: each byte of the buffer indicates the status of a 4K
        page (bit 0 set if read, bit 1 set if written)

(Table 03587)
Values for calling QEMM v5.00+ QPI function 1601h:
```

```
  AX = 1601h set memory access status
    ES:DI -> 256-byte buffer containing access statuses (see #03586)

(Table 03588)
Values for calling QEMM v5.00+ QPI function 17h:
  AH = 17h get memory usage statistics
    ES:DI -> 81-byte buffer for memory statistics (see #03645)
    Return: CF clear

(Table 03589)
Values for calling QEMM v5.11+ QPI function 18h:
  AH = 18h check whether conventional memory mapped into address range
    ES:BX = starting address
    CX = number of 4K pages
    Return: CF clear
      AL = 00h one or more pages is remapped
           01h all pages in range are conventional memory
        (physical address == virtual address)

(Table 03590)
Values for calling QEMM v5.11+ QPI function 19h:
  AH = 19h NOP
    Return: CF set

(Table 03591)
Values for calling QEMM v5.11+ "QPI_UntrappedIORead" function:
  AX = 1A00h get byte from I/O port
  DX = port number
Return: CF clear
  BL = port value
Note: this function was officially documented with the release of QEMM 7.50

(Table 03592)
Values for calling QEMM v5.11+ "QPI_UntrappedIOWrite" function:
  AX = 1A01h send byte to I/O port
  BL = value to send
  DX = port number
Return: CF clear
Note: this function was officially documented with the release of QEMM 7.50

(Table 03593)
```

```
Values for calling QEMM v5.11+ "QPI_UntrappedIOReadIndexed" function:
  AX = 1A02h
  BH = index value to send
  DX = base port number
Return: CF clear
  BL = value read from I/O port (DX+1)
Note: this function was officially documented with the release of QEMM 7.50

(Table 03594)
Values for calling QEMM v5.11+ "QPI_UntrappedIOWriteIndexed" function:
  AX = 1A03h send bytes to two consecutive I/O ports
  BH = value for first I/O port (DX)
  BL = value for second I/O port (DX+1)
  DX = base port number
Return: CF clear
Note: this function was officially documented with the release of QEMM 7.50

(Table 03595)
Values for calling QEMM v7.03+ "QPI_UntrappedIO" function:
  AX = 1A04h
  BX = value to write to port
  CX = direction and size
       bit 2: output instead of input
       bit 3: word instead of byte
  DX = I/O port to be accessed
Return: CF clear
  BX = value read (if CX indicates read)
Note: this function was officially documented with the release of QEMM 7.50

(Table 03596)
Values for calling QEMM v7.03+ function 1A05h
  AX = 1A05h
  ???
Return: ???

(Table 03597)
Values for calling QEMM v7.03+ "QPI_GetIOCallback" function:
  AX = 1A06h
Return: CF clear
  ES:DI -> current I/O callback function (see #03599)
Note: this function was officially documented with the release of QEMM 7.50
```

```
(Table 03598)
Values for calling QEMM v7.03+ "QPI_SetIOCallback" function:
  AX = 1A07h
  ES:DI -> new I/O callback function (see #03599)
Return: CF clear
Note: this function was officially documented with the release of QEMM 7.50

(Table 03599)
Values QEMM v7.03+ I/O callback function is called with:
  AL/AX = data to/from trapped port
  CL = I/O direction (00h = IN instruction, else OUT instruction)
  DX = I/O port address
Return: CF clear if port handled by callback function
  CF set if not handled
  all other registers returned to application executing the IN or OUT
    instruction (allowing arbitrary changes to port address, data value,
    etc.)
SeeAlso: #03597,#03598

(Table 03600)
Values for calling QEMM v7.03+ "QPI_GetPortTrap" function:
  AX = 1A08h
  DX = I/O port number
Return: CF clear
  BL = trapping state (00h not being trapped, 01h trap installed)
Note: this function was officially documented with the release of QEMM 7.50

(Table 03601)
Values for calling QEMM v7.03+ "QPI_SetPortTrap" function:
  AX = 1A09h
  DX = I/O port number
Return: CF clear
Note: this function was officially documented with the release of QEMM 7.50

(Table 03602)
Values for calling QEMM v7.03+ "QPI_ClearPortTrap" function:
  AX = 1A0Ah
  DX = I/O port number
Return: CF clear
Note: this function was officially documented with the release of QEMM 7.50
```

```
(Table 03603)
Values for calling QEMM v5.11+ QPI function 1B00h:
  AX = 1B00h get EMM Import Structure address
  ES:DI -> buffer for EMM import data structure (see #03643)
Return: CF set on error
  CF clear if successful
SeeAlso: INT 21/AX=4402h/SF=01h

(Table 03604)
Values for calling QEMM v5.11+ QPI function 1B01h:
  AX = 1B01h disable V86 mode
Return: CF set on error
      (i.e. no Global EMM Import rec. allocated)
  CF clear if successful
Note: shuts down EMS and initializes Global EMM Import record; this function
    is invoked from the callback supplied by INT 2F/AX=1605h

(Table 03605)
Values for calling QEMM v5.11+ QPI function 1B02h:
  AX = 1B02h enable V86 mode
Return: CF set on error
  CF clear if successful
Note: restarts EMS and frees Global EMM Import record; this function is
    invoked from the callback supplied by INT 2F/AX=1605h

(Table 03606)
Values for calling QEMM v5.11+ QPI function 1B03h:
  AX = 1B03h MS Windows initializing
  CX = segment from which Windows init broadcast made???
  DX = Windows startup flags
  DI = Windows version number (major in upper byte)
Return: CF clear if successful
      DS:SI -> V86 mode enable/disable callback
      (see #02634 at INT 2F/AX=1605h)
      ES:BX -> startup info structure (see #02631 at INT 2F/AX=1605h)
  CF set on error (unable to start Windows)
SeeAlso: INT 2F/AX=1605h

(Table 03607)
Values for calling QEMM v5.11+ QPI function 1B04h:
```

```
  AX = 1B04h MS Windows terminating
Return: CF clear

(Table 03608)
Values for calling QEMM v5.11+ QPI function 1B05h:
  AX = 1B05h determine whether program is driver
  DS:DX -> ASCIZ filename
Return: CF clear
  AL = status
       01h if string ends in ".DRV"
       FFh if string ends in "GDI.EXE"
       00h otherwise
Note: when MS Windows 3.0 standard mode starts, QEMM patches all drivers
    until GDI.EXE is loaded

(Table 03609)
Values for calling QEMM v5.11+ QPI function 1B06h:
  AX = 1B06h patch protected-mode check in Windows driver
  CX = length of data pointed at by DS:DX
  DS:DX -> buffer containing Windows driver code
Return: CF clear
Note: patches all SMSW x/TEST x,1 instruction sequences into MOV x,CS/VERW x
    sequences, which has the effect that the protected-mode check will
    only indicate protected mode in native protected mode and not in V86
    mode

(Table 03610)
Values for calling QEMM v5.11+ QPI function 1B07h:
  AX = 1B07h
BUG: QEMM 6.00-7.01 accept this and branch randomly

(Table 03611)
Values for calling QEMM v5.11+ QPI function 1Bxxh:
  AX = 1B08h to 1BFFh
Return: CF set

(Table 03612)
Values for calling QEMM v5.11+ QPI function 1C00h:
  AX = 1C00h disable IRQ0-7 calldowns
Return: CF clear
```

```
(Table 03613)
Values for calling QEMM v5.11+ QPI function 1C01h:
  AX = 1C01h set V86-mode IRQ0-7 handlers
  ES:DI -> 8 DWORDs containing V86-mode handler addresses
Return: CF clear

(Table 03614)
Values for calling QEMM v5.11+ QPI function 1C02h:
  AX = 1C02h disable IRQ8-15 handlers
Return: CF clear

(Table 03615)
Values for calling QEMM v5.11+ QPI function 1C03h:
  AX = 1C03h set V86-mode IRQ8-15 handlers
  ES:DI -> 8 DWORDs containing V86-mode handler addresses
BUG:  although the jump table only contains four entries, QEMM 6.00 will
    attempt to use it for any value of AL between 00h and 2Ah, thus
    branching unpredictably for AL=04h-2Ah; QEMM v7.01 behaves
    similarly for AL=04h-1Bh
Note: when enabled, the appropriate IRQs are reflected back to the specified
    handlers in virtual-86 mode after the CPU automatically invokes the
    protected-mode handler inside QEMM

(Table 03616)
Values for calling QEMM v7.03+ "QPI_SimulateHWInt" function:
  AX = 1C04h
  BX = number of interrupt to simulate
Return: ???
Notes:  this function will allow proper simulation of a hardware interrupt
    under DESQview and DESQview/X, where the correct interrupt handler
    may be in a different process with a completely different address
    space
  this function was officially documented with the release of QEMM v7.50

(Table 03617)
Values for calling QEMM v6.0x only QPI function 1D00h:
  AX = 1D00h switch to pre-Stealth interrupt vector table
Return: CF clear if supported (QEMM v6.x)
  CF set if not supported (QEMM v7+)
Notes:  also switches VGA Save table pointer (0040h:00A8h) and overwrites the
    vectors currently assigned for use by the two interrupt controllers
```

```
    (see INT 67/AX=DE0Ah) with the vectors for INT 08-0F and 70-77 (to
    avoid crashing the system).
  functions 1Dxxh are not supported by QEMM v7.01, and always return CF
    set

(Table 03618)
Values for calling QEMM v6.0x only QPI function 1D01h:
  AX = 1D01h restore user interrupt vector table
Return: CF clear if supported (QEMM v6.x)
  CF set if not supported (QEMM v7+)
Notes:  interrupts should be disabled around the AX=1D00h and AX=1D01h calls
    because QEMM does not modify the memory maps to map in ROM, so
    an interrupt could be disastrous
  clears any pending IRQ7 at end of function
  functions 1Dxxh are not supported by QEMM v7.01, and always return CF
    set

(Table 03619)
Values for calling QEMM v6.00+ QPI function 1Dxxh:
  AX = 1D02h to 1DFFh
Return: CF set

(Table 03620)
Values for calling QEMM v6.00+ "QEMM_GET_INFO"/"QPI_GetInfo" function:
  AX = 1E00h get Stealth configuration
Return: CF clear
  BL = memory configuration flags (documented as "reserved") (see #03644)
  BH = (v7.00+) disk buffer flags
       bit 0: DISKBUFFRAME buffer instead of DISKBUF buffer
       bit 1: buffer has already been used
  CL = stealth type (00h none, 46h Frame, 4Dh Map, 50h Protect)
  CH = suspend/resume interrupt (00h none)
  DL = (v7.00+) size of QEMM disk buffer in KB (00h none)
  DH = reserved (always 00h for v6.00)
  SI = reserved (always 0000h for v6.00)
  DI = reserved (always 0000h for v6.00)
Note: this function is officially documented

(Table 03621)
Values for calling QEMM v6.00+ "QPI_GetStealthCount" function:
  AX = 1E01h get number of Stealth'ed ROMs
```

```
Return: CF clear
  BX = number of Stealth'ed ROMs
Note: this function is officially documented

(Table 03622)
Values for calling QEMM v6.00+ "QPI_GetStealthList" function:
  AX = 1E02h
  ES:DI -> buffer for Stealth ROM info (see #03646)
Return: CF clear
  BX = number of Stealth'ed ROMs
  ES:DI buffer filled
Note: this function is officially documented

(Table 03623)
Values for unimplemented Stealth information functions:
  AX = 1E03h to 1EFFh
Return: CF set


(Table 03624)
Values for calling QEMM v6.00+ "QEMM_GET_PTE"/"QPI_GetPTE" function:
  AX = 1F00h get page table entry
  CX = page number (0000h-010Fh)
Return: CF clear
  EDX = page table entry
Note: this function is officially documented

(Table 03625)
Values for calling QEMM v6.00+ "QEMM_SET_PTE"/"QPI_SetPTE" function:
  AX = 1F01h set page table entry
  CX = page number (0000h-010Fh)
  EDX = new page table entry
Return: CF clear
Note: this function is officially documented
SeeAlso: #03549

(Table 03626)
Values for calling QEMM v6.00+ QPI function 1Fxxh:
  AX = 1F02h to 1FFFh
Return: CF set
```

```
(Table 03627)
Values for calling QEMM v6.00+ "QEMM_GET_VHI_INFO"/"QPI_GetVHIInfo" function:
  AX = 2000h "QEMM_GET_VHI_INFO" get VirtualHDIRQ information
Return: CF clear
  BL = flags
       bit 7: VirtualHDIRQ setting respected
       (set if Stealth active)
       bits 6-1 reserved
       bit 0: VirtualHDIRQ currently enabled
     (INT 15/AH=90h suppressed when enabled)
Note: this function is officially documented
SeeAlso: #03628

(Table 03628)
Values for calling QEMM v6.00+ "QEMM_SET_VHI_INFO"/"QPI_SetVHIInfo" function:
  AX = 2001h set VirtualHDIRQ state
  BL bit 0 = new VirtualHDIRQ state
Return: CF clear
  BL = old VHI setting (bits 0 and 7, see #03627)
Note: this function is officially documented
SeeAlso: #03627

(Table 03629)
Values for calling QEMM v6.00+ QPI function 20xxh:
  AX = 2002h to 20FFh
Return: CF set

(Table 03630)
Values for calling QEMM v6.00+ "QEMM_COPY_STEALTH_ROMS"/"QPI_CopyStealthRoms":
  AX = 2100h copy data from Stealthed address space
  DS:SI -> start address of hidden memory to copy
  ES:DI -> buffer for copied data
  ECX = number of bytes to copy
Return: CF clear if successful
  CF set on error (no Stealth or DS:SI < C000h:0000h or DS:SI + ECX > 1M)
Note: this function was officially documented with the release of QEMM 7.50

(Table 03631)
Values for calling QEMM v6.00+ QPI function 21xxh:
  AX = 2101h to 21FFh
Return: CF set
```

```
(Table 03632)
Values for calling QEMM v6.03+ QPI function 2200h:
  AX = 2200h DESQview/X support -- get ???
Return: CF clear
  ES:DI -> ???

(Table 03633)
Values for calling QEMM v6.03+ QPI function 2201h:
  AX = 2201h DESQview/X support -- set ???
  ES:DI -> ??? or 0000h:0000h
Return: CF clear if successful
  CF set on error

(Table 03634)
Values for calling QEMM v6.04+ QPI function 2300h:
  AX = 2300h get ???
  BX = which ??? to get (must be 0000h for v6.04)
Return: CF clear if successful
      ES:DI -> ???
  CF set on error

(Table 03635)
Values for calling QEMM v6.04+ QPI function 2301h:
  AX = 2301h set ???
  BX = which ??? to set (must be 0000h for v6.04)
  ES:DI -> ???
Return: CF clear if successful
  CF set on error

(Table 03636)
Values for calling QEMM v6.04+ QPI function 2302h:
  AX = 2302h clear specified ???
  BX = which ??? to clear (must be 0000h for v6.04)
Return: CF clear if successful
  CF set on error

(Table 03637)
Values for calling QEMM v6.04+ QPI function 23FFh:
  AX = 23FFh clear all ???
Return: CF clear if successful
```

```
  CF set on error

(Table 03638)
Values for calling QEMM v6.04+ QPI function 23xxh:
  AX = 2303h to 23FEh
Return: CF set

(Table 03639)
Values for calling QEMM v7.01+ QPI function 24h:
  AH = 24h ST-DBL support
  AL = subfunction
       00h set ???
    EDX -> information table (EDX = segment SHL 16 + offset)
       01h ???
Return: CF clear if successful
  CF set on error

(Table 03640)
Values for calling QEMM unimplemented QPI functions:
  AH = 25h to FFh
Return: CF set

Format of QEMM protection level configuration:
Offset  Size  Description (Table 03641)
 00h  WORD  segment of 128 breakpoint (INT 3) instructions for use in
       DESQview protection level 3 interrupt vector checking, or
       0000h to disable; in pl3, INTs 00-7F are pointed at these
       breakpoints
 02h  DWORD -> array of actual interrupt handler addresses for INT 00-7F
       when interrupt vectors are pointed at protection level 3
       breakpoints
 06h  DWORD far pointer to ??? region list (see #03642)
 0Ah  DWORD far pointer to buffer for returned ???
 0Eh  DWORD seg:ofs of function to call on protection violation???
 12h  WORD  segment of ???
 14h  DWORD far pointer to DWORD containing number of paragraphs of
       ??? for segment at offset 12h
  ???

Format of protection level Region List:
Offset  Size  Description (Table 03642)
```

```
 00h  WORD  number of PAIRS of pointers to follow
 02h 2N DWORDs  start/end seg:ofs addresses of ??? regions
Note: QEMM converts the segmented addresses into linear addresses in place

Format of EMM Import structure:
Offset  Size  Description (Table 03643)
 00h  DWORD physical address of EMM import struct
 04h  BYTE  major version (v6.00 sets to 01h)
 05h  BYTE  minor version (v6.00 sets to 00h/0Bh)
SeeAlso: INT 21/AX=4402h/SF=01h

Bitfields for memory configuration flags:
Bit(s)  Description (Table 03644)
 0  conventional memory sorted
 1  conventional memory filled
 2  ???
 3  ???
 4  expanded memory is in use
 5  ???

Format of QEMM 6.0 memory statistics:
Offset  Size  Description (Table 03645)
 00h  BYTE  01h if Shadow RAM found, 00h otherwise
 01h  DWORD initial conventional memory in bytes
 05h  DWORD initial extended memory in bytes
 09h  DWORD initial expanded memory in bytes
 0Dh  DWORD initial "top" or "shadow" memory in bytes
 11h  DWORD Unavailable conventional memory in bytes
 15h  DWORD Unavailable extended memory in bytes
 19h  DWORD Unavailable expanded memory in bytes
 1Dh  DWORD Unavailable "top" or "shadow" memory in bytes
    Add to offset 49h for Total unavailable top/shadow.
 21h  DWORD QEMM code size in bytes
 25h  DWORD QEMM data size in bytes
 29h  DWORD bytes used for TASKS=
 2Dh  DWORD DMA buffer size
 31h  DWORD bytes used for MAPS=
 35h  DWORD bytes of high RAM
 39h  DWORD bytes used by mapped ROMs
 3Dh  DWORD bytes of conventional memory provided by QEMM
 41h  DWORD bytes of extended memory NOT converted by QEMM (EXT=xxx)
```

```
 45h  DWORD bytes of EMS/XMS pool memory provided by QEMM
 49h  DWORD Unavailable "top" or "shadow" memory in bytes
    Add to offset 1Dh for Total unavailable top/shadow.
 4Dh  DWORD conventional memory overhead in bytes
    (set to 0 by QEMM.COM prior to call)

Format of Stealth ROM info [array]:
Offset  Size  Description (Table 03646)
 00h  WORD  starting segment of ROM
 02h  WORD  length of ROM in paragraphs

Format of QEMM EGA/VGA DAC register virtualization buffer:
Offset  Size  Description (Table 03647)
 00h  BYTE  (temp) current color register number
 01h  BYTE  (temp) number of bytes written so far for current color reg
 02h 768 BYTEs  three bytes per color register
--------m-6740-------------------------------
INT 67 - LIM EMS - GET MANAGER STATUS
  AH = 40h
Return: AH = status (00h,80h,81h,84h) (see #03648)
Note: this call can be used only after establishing that the EMS driver is in
    fact present
SeeAlso: AH=3Fh,AX=FFA5h,@xxxxh:xxxxh"PMM"

(Table 03648)
Values for EMS function status:
 00h  successful
 80h  internal error
 81h  hardware malfunction
 82h  busy -- retry later
 83h  invalid handle
 84h  undefined function requested by application
 85h  no more handles available
 86h  error in save or restore of mapping context
 87h  insufficient memory pages in system
 88h  insufficient memory pages available
 89h  zero pages requested
 8Ah  invalid logical page number encountered
 8Bh  invalid physical page number encountered
 8Ch  page-mapping hardware state save area is full
 8Dh  save of mapping context failed
```

```
 8Eh  restore of mapping context failed
 8Fh  undefined subfunction
 90h  undefined attribute type
 91h  feature not supported
 92h  successful, but a portion of the source region has been overwritten
 93h  length of source or destination region exceeds length of region
    allocated to either source or destination handle
 94h  conventional and expanded memory regions overlap
 95h  offset within logical page exceeds size of logical page
 96h  region length exceeds 1M
 97h  source and destination EMS regions have same handle and overlap
 98h  memory source or destination type undefined
 9Ah  specified alternate map register or DMA register set not supported
 9Bh  all alternate map register or DMA register sets currently allocated
 9Ch  alternate map register or DMA register sets not supported
 9Dh  undefined or unallocated alternate map register or DMA register set
 9Eh  dedicated DMA channels not supported
 9Fh  specified dedicated DMA channel not supported
 A0h  no such handle name
 A1h  a handle found had no name, or duplicate handle name
 A2h  attempted to wrap around 1M conventional address space
 A3h  source array corrupted
 A4h  operating system denied access
--------m-6741-------------------------------
INT 67 - LIM EMS - GET PAGE FRAME SEGMENT
  AH = 41h
Return: AH = status (see also AH=40h)
      00h function successful
    BX = segment of page frame
SeeAlso: AH=58h,AH=68h
--------m-6742-------------------------------
INT 67 - LIM EMS - GET NUMBER OF PAGES
  AH = 42h
Return: AH = status (see also AH=40h)
      00h function successful
    BX = number of unallocated pages
    DX = total number of pages
BUG:  DOS 6.0 EMM386.EXE causes a system lock-up or reboot if in AUTO mode
    when this call is made; use AH=46h to ensure that EMM386 is ON
    before making this call
SeeAlso: INT 2F/AX=2702h
```

```
--------m-6743------------------------------
INT 67 - LIM EMS - GET HANDLE AND ALLOCATE MEMORY
  AH = 43h
  BX = number of logical pages to allocate
Return: AH = status (00h,80h,81h,84h,85h,87h,88h,89h) (see #03648)
  DX = handle if AH=00h
SeeAlso: AH=45h
--------u-674321------------------------------
INT 67 - Q87 v4+ - INSTALLATION CHECK
  AX = 4321h
  EAX = 87654321h (entire EAX value required)
Return: EAX = 12345678h if Q87 is installed
Note: this call requires that VCPI services be available; if they are not,
    this call will not be recognized
BUG:  v4.00-v4.03 will hang on most machines when run under bare DOS with
    no memory manager, because neither Q87 nor UNLOAD87 checks whether
    there is a valid INT 67 handler before performing an installation
    check via INT 67
SeeAlso: AX=2833h,INT 21/AX=4321h
--------m-6744------------------------------
INT 67 - LIM EMS - MAP MEMORY
  AH = 44h
  AL = physical page number (0-3)
  BX = logical page number
      or FFFFh to unmap (QEMM)
  DX = handle
Return: AH = status (00h,80h,81h,83h,84h,8Ah,8Bh) (see #03648)
SeeAlso: AH=69h
--------m-6745------------------------------
INT 67 - LIM EMS - RELEASE HANDLE AND MEMORY
  AH = 45h
  DX = EMM handle
Return: AH = status (00h,80h,81h,83h,84h,86h) (see #03648)
SeeAlso: AH=43h
--------m-6746------------------------------
INT 67 - LIM EMS - GET EMM VERSION
  AH = 46h
Return: AH = status (00h,80h,81h,84h) (see #03648)
  AL = EMM version number if AH=00h
--------m-6747------------------------------
INT 67 - LIM EMS - SAVE MAPPING CONTEXT
```

```
  AH = 47h
  DX = handle
Return: AH = status (00h,80h,81h,83h,84h,8Ch-8Eh) (see #03648)
SeeAlso: AH=48h
--------m-6748--------------------------------
INT 67 - LIM EMS - RESTORE MAPPING CONTEXT
  AH = 48h
  DX = handle
Return: AH = status (00h,80h,81h,83h,84h,8Eh) (see #03648)
SeeAlso: AH=47h
--------m-6749--------------------------------
INT 67 - LIM EMS - reserved - GET I/O PORT ADDRESSES
  AH = 49h
Note: defined in EMS 3.0, but undocumented in EMS 3.2
--------m-674A--------------------------------
INT 67 - LIM EMS - reserved - GET TRANSLATION ARRAY
  AH = 4Ah
Note: defined in EMS 3.0, but undocumented in EMS 3.2
--------m-674B--------------------------------
INT 67 - LIM EMS - GET NUMBER OF EMM HANDLES
  AH = 4Bh
Return: AH = status (00h,80h,81h,83h,84h) (see #03648)
  BX = number of EMM handles if AH=00h
--------m-674C--------------------------------
INT 67 - LIM EMS - GET PAGES OWNED BY HANDLE
  AH = 4Ch
  DX = EMM handle
Return: AH = status (see #02785)
  BX = number of logical pages if AH=00h
SeeAlso: AH=4Dh
--------m-674D--------------------------------
INT 67 - LIM EMS - GET PAGES FOR ALL HANDLES
  AH = 4Dh
  ES:DI -> array to receive information
Return: AH = status (00h,80h,81h,84h) (see #03648)
  ---if AH=00h---
  BX = number of active EMM handles
  array filled with 2-word entries, consisting of a handle and the
    number of pages allocated to that handle
SeeAlso: AH=4Ch
--------m-674E--------------------------------
```

```
INT 67 - LIM EMS - GET OR SET PAGE MAP
  AH = 4Eh
  AL = subfunction
       00h get mapping registers
       01h set mapping registers
       02h get and set mapping registers at once
       03h get size of page-mapping array
  DS:SI -> array holding information (AL=01h/02h)
  ES:DI -> array to receive information (AL=00h/02h)
Return: AH = status (00h,80h,81h,84h,8Fh,A3h) (see also AH=40h)
       00h successful
     AL = bytes in page-mapping array (AL=03h only)
     array pointed to by ES:DI receives mapping info (AL=00h/02h)
Notes:  this function was designed to be used by multitasking operating systems
     and should not ordinarily be used by appplication software.
  MD386 returns the size of the page-mapping array in AX instead of AL
SeeAlso: AH=4Fh
--------m-674F--------------------------------
INT 67 - LIM EMS 4.0 - GET/SET PARTIAL PAGE MAP
  AH = 4Fh
  AL = subfunction
       00h get partial page map
          DS:SI -> structure containing list of segments whose mapping
       contexts are to be saved
          ES:DI -> array to receive page map
       01h set partial page map
          DS:SI -> structure containing saved partial page map
       02h get size of partial page map
          BX = number of mappable segments in the partial map to be saved
Return: AH = status (00h,80h,81h,84h,8Bh,8Fh,A3h) (see also AH=40h)
       8Bh one of the specified segments is not mappable
       A3h contents of partial page map corrupted or count of mappable
     segments exceeds total number of mappable segments in system
  AL = size of partial page map for subfunction 02h
SeeAlso: AH=4Eh
--------m-6750--------------------------------
INT 67 - LIM EMS 4.0 - MAP/UNMAP MULTIPLE HANDLE PAGES
  AH = 50h
  AL = subfunction
       00h use physical page numbers
       01h use segment addresses
```

```
  DX = handle
  CX = number of entries in array
  DS:SI -> mapping array (see #03649)
Return: AH = status (00h,80h,81h,83h,84h,8Ah,8Bh,8Fh) (see #03648)
SeeAlso: AH=40h

Format of EMS mapping array entry:
Offset  Size  Description (Table 03649)
 00h  WORD  logical page number or FFFFh to unmap physical page
 02h  WORD  physical page number or segment address
--------m-6751-------------------------------
INT 67 - LIM EMS 4.0 - REALLOCATE PAGES
  AH = 51h
  DX = handle
  BX = number of pages to be allocated to handle
Return: AH = status (00h,80h,81h,83h,84h,87h,88h) (see #03650)
  BX = actual number of pages allocated to handle

(Table 03650)
Values for EMS function status:
 00h  successful
 80h  internal error
 81h  hardware malfunction
 83h  invalid handle
 84h  undefined function requested
 87h  more pages requested than present in system
 88h  more pages requested than currently available
 8Ah  invalid logical page number encountered
 8Bh  invalid physical page number encountered
 8Fh  undefined subfunction
 90h  undefined attribute type
 91h  feature not supported
 A0h  no such handle name
 A1h  duplicate handle name
--------m-6752-------------------------------
INT 67 - LIM EMS 4.0 - GET/SET HANDLE ATTRIBUTES
  AH = 52h
  AL = subfunction
       00h get handle attributes
    Return: AL = attribute
            00h handle is volatile
```

```
          01h handle is nonvolatile
      01h set handle attributes
    BL = new attribute
       00h handle is volatile
       01h handle is nonvolatile
      02h get attribute capability
    Return: AL = attribute capability
          00h only volatile handles supported
          01h both volatile and non-volatile supported
  DX = handle
Return: AH = status (00h,80h,81h,83h,84h,8Fh-91h) (see #03648)
SeeAlso: AH=53h
--------m-6753-------------------------------
INT 67 - LIM EMS 4.0 - GET/SET HANDLE NAME
  AH = 53h
  AL = subfunction
      00h get handle name
         ES:DI -> 8-byte buffer for handle name
      01h set handle name
         DS:SI -> 8-byte handle name
  DX = handle
Return: AH = status (00h,80h,81h,83h,84h,8Fh,A1h) (see #03648)
SeeAlso: AH=52h
--------m-6754-------------------------------
INT 67 - LIM EMS 4.0 - GET HANDLE DIRECTORY
  AH = 54h
  AL = subfunction
      00h get handle directory
         ES:DI -> buffer for handle directory (see #03651)
      01h search for named handle
         DS:SI -> 8-byte name
      02h get total number of handles
Return: AL = number of entries in handle directory (subfunction 00h)
  DX = value of named handle (subfunction 01h)
  BX = total number of handles (subfunction 02h)
  AH = status (00h,80h,81h,84h,8Fh,A0h,A1h) (see also #03650)
      A1h a handle found had no name

Format of EMS handle directory entry:
Offset  Size  Description (Table 03651)
 00h  WORD  handle
```

```
 02h  8 BYTEs handle's name
--------m-6755--------------------------------
INT 67 - LIM EMS 4.0 - ALTER PAGE MAP AND JUMP
  AH = 55h
  AL = subfunction
       00h physical page numbers provided by caller
       01h segment addresses provided by caller
  DX = handle
  DS:SI -> structure containing map and jump address
Return: (at target address unless error)
  AH = status (00h,80h,81h,83h,84h,8Ah,8Bh,8Fh) (see #03648)
SeeAlso: AH=56h
--------m-6756--------------------------------
INT 67 - LIM EMS 4.0 - ALTER PAGE MAP AND CALL
  AH = 56h
  AL = subfunction
       00h physical page numbers provided by caller
    DX = handle
    DS:SI -> structure containing page map and call address
       01h segment addresses provided by caller
    DX = handle
    DS:SI -> structure containing page map and call address
       02h get page map stack space required
    Return: BX = stack space required
Return: (if successful, the target address is called.  Use a RETF to return
   and restore mapping context)
  AH = status (00h,80h,81h,83h,84h,8Ah,8Bh,8Fh) (see #03648)
SeeAlso: AH=55h
--------m-6756FF------------------------------
INT 67 - RM386 v6.00 - ???
  AX = 56FFh
  DS:SI -> ???
  ???
Return: ???
--------m-6757--------------------------------
INT 67 - LIM EMS 4.0 - MOVE/EXCHANGE MEMORY REGION
  AH = 57h
  AL = subfunction
       00h move memory region
       01h exchange memory region
  DS:SI -> structure describing source and destination (see #03653)
```

```
Return: AH = status (see #03652)
Note: source and destination may overlap for a move, in which case the copy
    direction is chosen such that the destination receives an intact copy
    of the source region

(Table 03652)
Values for EMS function status:
 00h  successful
 80h  internal error
 81h  hardware failure
 83h  invalid handle
 84h  undefined function requested
 8Ah  invalid logical page number encountered
 8Fh  undefined subfunction
 92h  successful, but a portion of the source region has been overwritten
 93h  length of source or destination region exceeds length of region
    allocated to either source or destination handle
 94h  conventional and expanded memory regions overlap
 95h  offset within logical page exceeds size of logical page
 96h  region length exceeds 1M
 97h  source and destination EMS regions have same handle and overlap
 98h  memory source or destination type undefined
 A2h  attempted to wrap around 1M conventional address space

Format of EMS copy data:
Offset  Size  Description (Table 03653)
 00h  DWORD region length in bytes
 04h  BYTE  source memory type
    00h conventional
    01h expanded
 05h  WORD  source handle (0000h if conventional memory)
 07h  WORD  source initial offset (within page if EMS, segment if convent)
 09h  WORD  source initial segment (conv mem) or logical page (EMS)
 0Bh  BYTE  destination memory type
    00h conventional
    01h expanded
 0Ch  WORD  destination handle
 0Eh  WORD  destination initial offset
 10h  WORD  destination initial segment or page
--------m-6758-------------------------------
INT 67 - LIM EMS 4.0 - GET MAPPABLE PHYSICAL ADDRESS ARRAY
```

```
  AH = 58h
  AL = subfunction
       00h get mappable physical address array
     ES:DI -> buffer to be filled with array (see #03654)
       01h get number of entries in m.p.a. array
Return: CX = number of entries in array
  AH = status (00h,80h,81h,84h,8Fh) (see #03652)
Note: the returned array for subfunction 00h is filled in physical segment
    address order

Format of EMS mappable physical address entry:
Offset  Size  Description (Table 03654)
 00h  WORD  physical page segment
 02h  WORD  physical page number
--------m-675857-----------------------------
INT 67 U - NETROOM??? - ???
  AX = 5857h
  BX = function??? (0057h,0059h,0159h seen)
  ???
Return: ???
Note: BX=0059h appears to be analogous to AX=5800h and BX=0159h appears to
    be analogous to AX=5801h; BX=0057h appears to indicate whether
    AX=580xh or AX=5857h/BX=0x59h should be used
SeeAlso: AX=5BF0h
--------m-6759--------------------------------
INT 67 - LIM EMS 4.0 - GET EXPANDED MEMORY HARDWARE INFORMATION
  AH = 59h
  AL = subfunction
       00h get hardware configuration array
     ES:DI -> buffer to be filled with array (see #03655)
       01h get unallocated raw page count
     Return: BX = unallocated raw pages
       DX = total raw pages
Return: AH = status (see also AH=58h"EMS 4.0")
       A4h access denied by operating system
Note: subfunction 00h is for use by operating systems only, and can be
    enabled or disabled at any time by the operating system

Format of EMS hardware configuration array:
Offset  Size  Description (Table 03655)
 00h  WORD  size of raw EMM pages in paragraphs
```

```
 02h  WORD  number of alternate register sets
 04h  WORD  size of mapping-context save area in bytes
 06h  WORD  number of register sets assignable to DMA
 08h  WORD  DMA operation type
    0000h DMA with alternate register sets
    0001h only one DMA register set
--------m-675A--------------------------------
INT 67 - LIM EMS 4.0 - ALLOCATE STANDARD/RAW PAGES
  AH = 5Ah
  AL = subfunction
       00h allocate standard pages
       01h allocate raw pages
  BX = number of pages to allocate
Return: DX = handle
  AH = status (00h,80h,81h,84h,85h,87h,88h,8Fh) (see #03648)
--------m-675B--------------------------------
INT 67 - LIM EMS 4.0 - ALTERNATE MAP REGISTER SET
  AH = 5Bh
  AL = subfunction
       00h get alternate map register set
    Return: BL = current active alternate map register set number
       ES:DI -> map register context save area if BL=00h
       01h set alternate map register set
    BL = new alternate map register set number
    ES:DI -> map register context save area if BL=0
       02h get alternate map save array size
    Return: DX = array size in bytes
       03h allocate alternate map register set
    Return: BL = number of map register set; 00h = not supported
       04h deallocate alternate map register set
    BL = number of alternate map register set
Return: AH = status (00h,80h,81h,84h,8Fh,9Ah-9Dh,A3h,A4h) (see #03656)
Note: this function is for use by operating systems only, and can be
    enabled or disabled at any time by the operating system

(Table 03656)
Values for EMS function status:
 00h  successful
 80h  internal error
 81h  hardware malfunction
 84h  undefined function requested
```

```
 8Fh  undefined subfunction
 9Ah  specified alternate map register or DMA register set not supported
 9Bh  all alternate map register or DMA register sets currently allocated
 9Ch  alternate map register or DMA register sets not supported
 9Dh  undefined or unallocated alternate map register/DMA register set
 9Eh  dedicated DMA channels not supported
 9Fh  specified dedicated DMA channel not supported
 A3h  source array corrupted
 A4h  operating system denied access
--------m-675B-------------------------------
INT 67 - LIM EMS 4.0 - ALTERNATE MAP REGISTER SET - DMA REGISTERS
  AH = 5Bh
  AL = subfunction
       05h allocate DMA register set
     Return: BL = DMA register set number, 00h if not supported
       06h enable DMA on alternate map register set
          BL = DMA register set number
          DL = DMA channel number
       07h disable DMA on alternate map register set
          BL = DMA register set number
       08h deallocate DMA register set
          BL = DMA register set number
Return: AH = status (00h,80h,81h,84h,8Fh,9Ah-9Fh,A3h,A4h) (see #03656)
Note: this function is for use by operating systems only, and can be
    enabled or disabled at any time by the operating system
--------m-675BE0-----------------------------
INT 67 - MICEMM v4D, RM386 - GET LINEAR ADDRESS OF MEMORY
  AX = 5BE0h
  ES:BX -> memory for which to get linear address
Return: AH = 00h
  CX:DX = linear address of physical memory corresponding to ES:BX
Program: RAM-MAN/386 is the memory manager included with Helix's Netroom;
    MICEMM is a memory manager for some Micronics motherboards
Note: this has been superceded by AX=DE06h, which should be used instead
SeeAlso: AX=5BF0h,AX=5BF1h,AX=DE06h
--------m-675BE1-----------------------------
INT 67 - RM386 v6.00+ - GET MEMORY MANAGER SIZE
  AX = 5BE1h
Return: AH = 00h
  CX = code and data size in bytes
  DX:BX = physical address of RM386 code
```

```
  DI:SI = total size of RM386 area including handle tables
  BP = number of additional pages (high DOS, etc.)
SeeAlso: AX=5BE0h,AX=5BE2h
--------m-675BE2----------------------------
INT 67 - RM386 v6.00+ - GET INTERRUPT VECTORS
  AX = 5BE2h
Return: DS:SI -> V86-mode table (see #03657)
  ES:BX -> ??? (undoc, middle of device driver interrupt routine!)
SeeAlso: AX=5BE0h,AX=5BE1h

Format of RM386 V86-mode table:
Offset  Size  Description (Table 03657)
 00h  DWORD original INT 13 vector
 04h  DWORD original INT 15 vector
 08h  DWORD original INT 19 vector
 0Ch  DWORD original INT 21 vector
 10h  DWORD original INT 4B vector
 14h  DWORD original INT 67 vector
--------m-675BF0----------------------------
INT 67 - MICEMM v4D, RM386 - INSTALLATION CHECK
  AX = 5BF0h
Return: AH = 00h if MICEMM or RM386 present
      BX = code segment of driver
Program: MICEMM is the Micronics Expanded Memory Manager; RM386 is the memory
    manager included in Helix Software's Netroom
SeeAlso: AH=3Fh,AX=5BE0h,AX=5BF1h
--------m-675BF1----------------------------
INT 67 - MICEMM v4D, RM386 - GET ADDRESS MAP
  AX = 5BF1h
  ES:BX -> 256-byte (MICEMM) or 512-byte (RM386) buffer for memory types
Return: AH = 00h
  ES:BX buffer filled (see #03658)
Note: each byte in the buffer specifies the type of a 4K page of memory
SeeAlso: AX=5BE0h,AX=5BF0h

(Table 03658)
Values for MICEMM/RM386 memory type:
 00h  unused (MICEMM), RAM/available (RM386)
 02h  DOS extension (XMS UMB)
 04h  shadowed ROM
 08h  mappable EMS
```

```
 10h  page frame
 20h  ROM
 40h  reserved (video memory, etc)
 80h  RAM (MICEMM), Windows UMB (RM386)
--------m-675BF2-----------------------------
INT 67 - RM386 - GET RM386 INTERNAL DATA
  AX = 5BF2h
  CX = size of buffer
  DS:SI -> buffer for internal data
  (documentation says ES:BX -> buffer, SI = offset within RM386)
Return: buffer filled
Note: the data returned by this function is release-specific
SeeAlso: AX=5BF0h
--------m-675BF3-----------------------------
INT 67 - RM386 - RETURN TO REAL MODE
  AX = 5BF3h
Return: nothing
Note: use AX=5DE0h instead of this functin
SeeAlso: AX=5BF0h,AX=5DE0h
--------m-675BF4-----------------------------
INT 67 - RM386 v6.00 - GET RM386 GLOBAL FLAGS
  AX = 5BF4h
Return: AH = 00h
  BX = global flags 1 (see #03659)
  CX = global flags 2 (see #03660)
  DX = global flags 3 (see #03661)
  SI = global flags 4 (see #03662)
SeeAlso: AX=5BF0h

Bitfields for RM386 global flags 1:
Bit(s)  Description (Table 03659)
 0-3  reserved
 4  V86 mode
 5  reserved
 6  80386 or higher CPU
 7,8  reserved
 9  A20 enabled at startup
 10 "HIGH_IO"
 11 ROM
 12 large frame
 13,14  reserved
```

```
 15 PS/2-style A20 control

Bitfields for RM386 global flags 2:
Bit(s)  Description (Table 03660)
 0  HMA in use
 1  XMS present
 2  using XMS driver memory
 3  HIGH (NEAT only)
 4-7  reserved
 8  NOBKTRAP
 9  NORESET
 10 ALTMAP
 11 NOFRAME
 12-15  reserved

Bitfields for RM386 global flags 3:
Bit(s)  Description (Table 03661)
 0  NOTEST
 1  NOEBDA
 2  Windows3 support
 3  system board mouse
 4  DISKBUF
 5  EBDALOW
 6  A20 global enable flag
 7  A20 flag
 8  EBDA moved to stub
 9  VXD file was found
 10 reserved
 11 NOBOOTMAP
 12 AUTO
 13 PS/2 machine
 14 Compaq ROM merge active
 15 NOHMA set

Bitfields for RM386 global flags 4:
Bit(s)  Description (Table 03662)
 0  "NOV8259" don't virtualize interrupt controller
 1  NOSCSI
 2  NOSCAN
 3  NOTR
 4  ALTBOOT
```

```
 5  NOCOMPQ
 6  KB2TRAP
 7  DESHADOW
 8  Video 7 VGA detected
 9  reserved
 10 NOVGA
 11 NOPS2
 12 DEBUG
 13 NOVKB
 14,15  reserved
--------m-675BF5------------------------------
INT 67 - RM386 v6.00 - GET RM386 EMS HANDLE COUNT
  AX = 5BF5h
Return: AH = status
      00h successful
    BX = current number of allocated EMS handles
      84h function not available
SeeAlso: AX=5BF0h
--------m-675C--------------------------------
INT 67 - LIM EMS 4.0 - PREPARE EXPANDED MEMORY HARDWARE FOR WARM BOOT
  AH = 5Ch
Return: AH = status (see #03663)
Note: when MS-DOS v6.xx EMM386 is loaded and the keyboard driver supports
    INT 15/AH=4Fh (keyboard intercept) calls, the system may hang
    instead of booting if this function is called just prior to a jump
    to F000h:FFF0h

(Table 03663)
Values for EMS function status:
 00h  successful
 80h  internal error
 81h  hardware malfunction
 84h  undefined function requested
--------m-675D--------------------------------
INT 67 - LIM EMS 4.0 - ENABLE/DISABLE OS FUNCTION SET FUNCTIONS
  AH = 5Dh
  AL = subfunction
      00h enable OS Function Set
      01h disable OS Function Set
      02h return access key (resets memory manager, returns access key at
    next invocation)
```

```
  BX,CX = access key returned by first invocation
Return: BX,CX = access key, returned only on first invocation of function
  AH = status (see also AH=5Ch)
       8Fh undefined subfunction
       A4h operating system denied access
--------m-675D03----------------------------
INT 67 u - Nanosoft MD386 - INTERNAL INITIALIZATION
  AX = 5D03h
  ???
Return: ???
Program: MD386 is a subset EMS memory manager by Nanosoft specifically
    designed for use with the MultiDOS Plus multitasker
SeeAlso: AX=5D04h,AX=5E00h
--------m-675D04----------------------------
INT 67 - Nanosoft MD386 - GET ALTERNATE MAP STRUCTURE
  AX = 5D04h
  BX = alternate register set number
  ES:DI -> 1024-byte buffer for map structure
Return: AH = status (see #03648)
  buffer filled if AH=00h
Note: used for debugging purposes
SeeAlso: AX=5D05h
--------m-675D05----------------------------
INT 67 - Nanosoft MD386 - GET INTERNAL HANDLE TABLE
  AX = 5D05h
  BX = handle number
  ES:DI -> 1024-byte buffer for handle table
Return: AH = status (see #03648)
  buffer filled if AH=00h
Note: used for debugging purposes
SeeAlso: AX=5D04h
--------m-675DE0----------------------------
INT 67 - RM386 - DISABLE RM386
  AX = 5DE0h
Note: RM386 traps this function on the initial transition to protected
    mode caused by the INT instruction, which means it can not be
    overridden simply by hooking the interrupt
SeeAlso: AX=5DE1h
--------m-675DE1----------------------------
INT 67 - RM386 - ENABLE RM386
  AX = 5DE1h
```

```
Note: RM386 traps this function on the initial transition to protected
    mode caused by the INT instruction, which means it can not be
    overridden simply by hooking the interrupt
SeeAlso: AX=5DE0h
--------m-675DE2-----------------------------
INT 67 - RM386 - GET PAGE TABLE
  AX = 5DE2h
  ES:DI -> 1088-byte buffer for page table
Return: ES:DI buffer filled
Note: RM386 traps this function on the initial transition to protected
    mode caused by the INT instruction, which means it can not be
    overridden simply by hooking the interrupt
SeeAlso: AX=5DE3h
--------m-675DE3-----------------------------
INT 67 - RM386 - SET PAGE TABLE
  AX = 5DE3h
  ES:DI -> 1088-byte buffer containing page table
Notes:  only the access bits of the page table are used, the remainder is
    ignored
  RM386 traps this function on the initial transition to protected
    mode caused by the INT instruction, which means it can not be
    overridden simply by hooking the interrupt
SeeAlso: AX=5DE2h
--------m-675DE4-----------------------------
INT 67 - RM386 - SET WRITE-PROTECTION FOR PAGE IN FIRST MEGABYTE
  AX = 5DE4h
  BL = page number
  BH = access (00h read-only, 01h read-write)
Note: RM386 traps this function on the initial transition to protected
    mode caused by the INT instruction, which means it can not be
    overridden simply by hooking the interrupt
--------m-675DE5-----------------------------
INT 67 - RM386 - MAP PHYSICAL PAGE TO PHYSICAL SEGMENT
  AX = 5DE5h
  EBX = physical page number
  DX = page number in first megabyte to be remapped (linear-addr SHR 12)
Return: AH = status
      00h successful
      8Bh invalid destination page (not in first megabyte)
Note: RM386 traps this function on the initial transition to protected
    mode caused by the INT instruction, which means it can not be
```

```
    overridden simply by hooking the interrupt
SeeAlso: AX=5DE6h
--------m-675DE6-----------------------------
INT 67 - RM386 - MAP LOGICAL 4K PAGE TO PHYSICAL SEGMENT
  AX = 5DE6h
  BX = logical page number in 4K pages from beginning of memory for EMS
       handle
  CX = segment in first megabyte to be remapped
  DX = previously-allocated EMS handle
Return: AH = status
      00h successful
      83h invalid handle
      8Ah invalid logical page (out of handle's range)
      8Bh invalid destination page (not in first megabyte)
Note: RM386 traps this function on the initial transition to protected
    mode caused by the INT instruction, which means it can not be
    overridden simply by hooking the interrupt
SeeAlso: AX=5DE5h
--------m-675DE7-----------------------------
INT 67 - RM386 - SET PAGE TABLE BITS FOR RANGE OF PAGES
  AX = 5DE7h
  BL = page table bits to be set (bits 2-0 = U/S, R/W, P)
  CX = number of pages to set
  DX = first page number to set (in first megabyte)
Return: AH = status
      00h successful
      8Bh invalid destination page (not in first megabyte)
      A5h invalid page bits
      A6h invalid page count (overflows first megabyte)
Note: RM386 traps this function on the initial transition to protected
    mode caused by the INT instruction, which means it can not be
    overridden simply by hooking the interrupt
--------m-675DE8-----------------------------
INT 67 - RM386 - GET PARTIAL PAGE TABLE
  AX = 5DE8h
  BX = starting page number in first megabyte+HMA (0000h-010Fh)
  CX = number of page table entries to get
  ES:DI -> buffer for DWORD page table entries
Return: AH = status (00h successful, 8Bh invalid page)
Note: RM386 traps this function on the initial transition to protected
    mode caused by the INT instruction, which means it can not be
```

```
    overridden simply by hooking the interrupt
SeeAlso: AX=5DE9h
--------m-675DE9-----------------------------
INT 67 - RM386 - SET PARTIAL PAGE TABLE
  AX = 5DE9h
  BX = starting page number in first megabyte+HMA (0000h-010Fh)
  CX = number of page table entries to get
  DS:SI -> buffer of DWORD page table entries
Return: AH = status (00h successful, 8Bh invalid destination page)
Note: RM386 traps this function on the initial transition to protected
    mode caused by the INT instruction, which means it can not be
    overridden simply by hooking the interrupt
SeeAlso: AX=5DE8h
--------m-675DEA-----------------------------
INT 67 - RM386 - V86-MODE I/O PORT TRAPPING CONTROL
  AX = 5DEAh
  BX = function
       00h globally disable V86-mode trapping
       01h globally enable V86-mode trapping
    CL = interrupt to use for trapping
       02h get I/O trapping state
Return: AH = status
       00h successful
    BX = current trapping state (function 02h)
         0000h disabled, 0001h enabled
    CX = interrupt used as trap interrupt (functions 00h and 02h)
Notes:  RM386 traps this function on the initial transition to protected
    mode caused by the INT instruction, which means it can not be
    overridden simply by hooking the interrupt
  when I/O trapping is enabled and I/O port access occurs, RM386
    simulates an INT instruction for the specified interrupt; the
    interrupt handler is responsible for decoding the trapped instruction
    and performing the appropriate action.  INT 2C/AX=002Dh provides a
    similar but more-easily used interface.
SeeAlso: AX=5DEBh,AH=EFh"RM386",INT 2C/AX=002Dh
--------m-675DEB-----------------------------
INT 67 - RM386 - V86-MODE I/O TRAPPING PORT CONTROL
  AX = 5DEBh
  BX = function
       00h disable V86-mode trapping for specified port
       01h enable V86-mode trapping for specified port
```

```
      02h get V86-mode trapping state for specified port
  DX = port for which to enable/disable/query trapping
Return: AH = status
      00h successful
    BX = current trapping state (00h off, 01h on) (function 02)
      A7h invalid port ID
      A8h reserved port--cannot trap/untrap (DMA/INT/KBD controllers)
Notes:  RM386 traps this function on the initial transition to protected
    mode caused by the INT instruction, which means it can not be
    overridden simply by hooking the interrupt
SeeAlso: AX=5DEAh
--------m-675DFD-----------------------------
INT 67 U - RM386 v6.00 - ???
  AX = 5DFDh
  ???
Return: ???
Note: RM386 traps this function on the initial transition to protected
    mode caused by the INT instruction, which means it can not be
    overridden simply by hooking the interrupt
SeeAlso: AX=5DFEh
--------m-675DFE-----------------------------
INT 67 U - RM386 v6.00 - ???
  AX = 5DFEh
  ???
Return: ???
Note: RM386 traps this function on the initial transition to protected
    mode caused by the INT instruction, which means it can not be
    overridden simply by hooking the interrupt
SeeAlso: AX=5DFDh
--------m-675DFF-----------------------------
INT 67 U - RM386 v6.00 - ???
  AX = 5DFFh
  ???
Return: ???
Note: RM386 traps this function on the initial transition to protected
    mode caused by the INT instruction, which means it can not be
    overridden simply by hooking the interrupt
SeeAlso: AX=5DFDh,AX=5DFEh
--------m-675E00-----------------------------
INT 67 - Nanosoft MD386 - SET HARDWARE BREAKPOINT
  AX = 5E00h
```

```
  DH = breakpoint number (0-3)
  DL = breakpoint attributes (used to set DR7)
  CX:BX = linear address of breakpoint
SeeAlso: AX=5D03h,AX=5E01h
--------m-675E01-----------------------------
INT 67 - Nanosoft MD386 - GET HARDWARE DEBUG REGISTER
  AX = 5E01h
  BL = register number (0-3,7)
Return: CX:BX = value of specified DRx register
SeeAlso: AX=5E00h
--------m-675E02-----------------------------
INT 67 - Nanosoft MD386 - SET DEBUG EXCEPTION HANDLER
  AX = 5E02h
  CX:BX -> exception handler
Note: the specified exception handler is called with a simulated interrupt
    whenever a debug exception occurs which was caused by a hardware
    breakpoint set with the debug registers
SeeAlso: AX=5E04h,AX=5E05h
--------m-675E03-----------------------------
INT 67 - Nanosoft MD386 - ENABLE/DISABLE MEMORY WRITE PROTECTION
  AX = 5E03h
  BL = register map set number
  BH = new state (00h read-only, else read-write)
  CX = linear page number (linear address SHR 12)
Note: setting write protection in map set 0 will cause the setting to become
    the default for newly-allocated map sets
--------m-675E04-----------------------------
INT 67 - Nanosoft MD386 - GET DEBUG EXCEPTION
  AX = 5E04h
Return: BL = debug exception (low byte of DR6 register)
SeeAlso: AX=5E02h,AX=5E05h
--------m-675E05-----------------------------
INT 67 - Nanosoft MD386 - IGNORE NEXT DEBUG EXCEPTION
  AX = 5E05h
Note: may be required when using AX=5E02h for handling instruction
    breakpoints
SeeAlso: AX=5E02h,AX=5E04h
--------m-6760-------------------------------
INT 67 - EEMS - GET PHYSICAL WINDOW ARRAY
  AH = 60h
  ES:DI -> buffer
```

```
Return: AH = status (see also AH=40h)
  AL = number of entries
  buffer at ES:DI filled
--------m-6761-------------------------------
INT 67 - EEMS - GENERIC ACCELERATOR CARD SUPPORT
  AH = 61h
  ???
Return: ???
Note: can be used by accelerator card manufacturer to flush RAM cache,
    ensuring that the cache accurately reflects what the processor would
    see without the cache.
--------m-676100-----------------------------
INT 67 - EEMS - STB GENERIC ACCELERATOR CARD SUPPORT - INSTALLATION CHECK???
  AX = 6100h
Return: AH = status (00h if installed???)
Note: STB's RAPIDMAP.SYS EMS driver calls this function and AX=6101h if
    the signature "GACXXX" is found at offset 0Ah in the INT 67 handler's
    segment (i.e. a device driver named GACXXX?? has hooked INT 67)
SeeAlso: AX=6101h,INT 02/SI=0714h
--------m-676101-----------------------------
INT 67 - EEMS - STB GENERIC ACCELERATOR CARD SUPPORT - GET ??? ENTRY POINT
  AX = 6101h
Return: AH = status
  ---if AH=00h---
  ES:BX -> ??? entry point
SeeAlso: AX=6100h
--------m-6768-------------------------------
INT 67 - EEMS - GET ADDRESSES OF ALL PAGE FRAMES IN SYSTEM
  AH = 68h
  ES:DI -> buffer
Return: AH = status (see also AH=40h)
  AL = number of entries
  buffer at ES:DI filled
Note: equivalent to LIM 4.0 function 58h
--------m-6769-------------------------------
INT 67 - EEMS - MAP PAGE INTO FRAME
  AH = 69h
  AL = frame number
  BX = page number
  DX = handle
Return: AH = status (see also AH=40h)
```

```
Note: similar to EMS function 44h
SeeAlso: AH=44h,AH=50h,AH=6Ah
--------m-676A-------------------------------
INT 67 - EEMS - PAGE MAPPING
  AH = 6Ah
  AL = subfunction
       00h save partial page map
    CH = first page frame
    CL = number of frames
    ES:DI -> buffer which is to be filled
      01h restore partial page map
    CH = first page frame
    CL = number of frames
    DI:SI -> previously saved page map
      02h save and restore partial page map
    CH = first page frame
    CL = number of frames
    ES:DI = buffer for current page map
    DI:SI = new page map
      03h get size of save array
    CH = first page frame
    CL = number of frames
    Return: AL = size of array in bytes
      04h switch to standard map register setting
      05h switch to alternate map register setting
      06h deallocate pages mapped to frames in conventional memory
    CH = first page frame
    CL = number of frames
Return: AH = status (see #03648)
Note: similar to EMS function 4Eh, except that a subrange of pages can be
    specified
SeeAlso: AH=69h
--------m-676B-------------------------------
INT 67 - DESQview 2.42-2.53 - BUG
  AH = 6Bh
Note: the EMM.DVR portion of DESQview branches to a random location on this
    function due to a fencepost error
--------m-67DD-------------------------------
INT 67 - Quadtel QMAPS - API
  AH = DDh
  AL = function
```

```
  ???
Return: ???
Notes:  details are not yet available
  Hewlett-Packard's HPMM.SYS is a licensed version of QMAPS, and thus
    supports this API
SeeAlso: AH=3Fh,AX=FFA5h
--------E-67DE00-----------------------------
INT 67 - Virtual Control Program Interface - INSTALLATION CHECK
  AX = DE00h
Return: AH = status
      00h VCPI is present
    BH = major version number
    BL = minor version number
      nonzero  VCPI not present
BUG:  MS Windows 3.00 is reported to "object violently" to this call.
SeeAlso: INT 2F/AX=1687h
--------E-67DE01-----------------------------
INT 67 - Virtual Control Program Interface - GET PROTECTED MODE INTERFACE
  AX = DE01h
  ES:DI -> 4K page table buffer
  DS:SI -> three descriptor table entries in GDT
    first becomes code segment descriptor, other two for use by
      main control program
Return: AH = 00h successful
      DI -> first unused page table entry in buffer
      EBX -> protected mode entry point in code segment (see #03664)
  AH = nonzero  failed
Note: protected mode entry point may be called with AX=DE00h-DE05h and
    AX=DE0Ch (in each case, all other registers as appropriate for
    the function)
SeeAlso: INT 2F/AX=1687h,INT 67/AH=3Fh

(Table 03664)
Call QEMM v6.03 protected mode entry point additionally with:
  AX = DF00h ???
    ???
    Return: ???
  AX = DF01h ???
    ???
    Return: ???
--------E-67DE02-----------------------------
```

```
INT 67 - Virtual Control Program Interface - GET MAX PHYSICAL MEMORY ADDRESS
  AX = DE02h
Return: AH = 00h  successful
      EDX = physical address of highest 4K memory page
  AH nonzero: failed
SeeAlso: AH=3Fh
--------E-67DE03----------------------------
INT 67 - Virtual Control Program Interface - GET NUMBER OF FREE 4K PAGES
  AX = DE03h
Return: AH = 00h  successful
      EDX = number of free 4K pages
  AH nonzero: failed
Notes:  returns total number of pages available to ALL tasks in system
  also available in protected mode by calling the protected-mode VCPI
    entry point (see AX=DE01h,#03664)
SeeAlso: AX=DE04h
--------E-67DE04----------------------------
INT 67 - Virtual Control Program Interface - ALLOCATE A 4K PAGE
  AX = DE04h
Return: AH = 00h successful
      EDX = physical address of allocated page
  AH nonzero: failed
Notes:  the client program is responsible for freeing all memory allocated
    with this call before terminating
  also available in protected mode by calling the protected-mode VCPI
    entry point (see AX=DE01h,#03664)
SeeAlso: AH=3Fh,AX=DE03h,AX=DE05h
--------E-67DE05----------------------------
INT 67 - Virtual Control Program Interface - FREE 4K PAGE
  AX = DE05h
  EDX = physical address of 4K page
Return: AH = status
      00h successful
      nonzero failed
Note: also available in protected mode by calling the protected-mode VCPI
    entry point (see AX=DE01h,#03664)
SeeAlso: AH=3Fh,AX=DE04h
--------E-67DE06----------------------------
INT 67 - Virtual Control Program Interface - GET PHYS ADDR OF PAGE IN FIRST MB
  AX = DE06h
  CX = page number (linear address shifted right 12 bits)
```

```
Return: AH = status
      00h successful
    EDX = physical address of page
      nonzero invalid page number (AH = 8Bh recommended)
SeeAlso: AX=5BE0h
--------E-67DE07----------------------------
INT 67 - Virtual Control Program Interface - READ CR0
  AX = DE07h
Return: AH = 00h
  EBX = value of Control Register 0
SeeAlso: AH=3Fh,AX=DE07h
--------E-67DE08----------------------------
INT 67 - Virtual Control Program Interface - READ DEBUG REGISTERS
  AX = DE08h
  ES:DI -> array of 8 DWORDs
Return: AH = 00h
  buffer filled with DR0 first, DR7 last, DR4 and DR5 unused
SeeAlso: AH=3Fh,AX=DE09h
--------E-67DE09----------------------------
INT 67 - Virtual Control Program Interface - SET DEBUG REGISTERS
  AX = DE09h
  ES:DI -> array of 8 DWORDs holding new values of debug registers
Return: AH = 00h
Note: values for DR4 and DR5 ignored
SeeAlso: AH=3Fh,AX=DE08h
--------E-67DE0A----------------------------
INT 67 - Virtual Control Program Interface - GET 8259 INTERRUPT VECTOR MAPPINGS
  AX = DE0Ah
Return: AH = 00h successful
      BX = first vector used by master 8259 (IRQ0)
      CX = first vector used by slave 8259 (IRQ8)
  AH nonzero: failed
Note: CX is undefined in systems without a slave 8259
SeeAlso: AX=DE0Bh,INT 21/AX=250Ch,INT 31/AX=0400h
--------E-67DE0B----------------------------
INT 67 - Virtual Control Program Interface - SET 8259 INTERRUPT VECTOR MAPPINGS
  AX = DE0Bh
  BX = first vector used by master 8259
  CX = first vector used by slave 8259
  interrupts disabled
Return: AH = 00h successful
```

```
  AH nonzero: failed
Notes:  This call merely informs the server that the client has changed the
    interrupt mappings.  The client may not change the mappings if they
    have already been changed by the server or another client, and is
    responsible for restoring the original mappings before terminating.
SeeAlso: AX=DE0Ah,INT 2C/AX=002Ah
--------E-67DE0C----------------------------
INT 67 - Virtual Control Program Interface - SWITCH TO PROTECTED MODE
  AX = DE0Ch
  ESI = linear address in first megabyte of values for system
        registers (see #03665)
  interrupts disabled
Return: interrupts disabled
  GDTR, IDTR, LDTR, TR loaded
  SS:ESP must have at least 16 bytes space, and the entry point is
        required to set up a new stack before enabling interrupts
  EAX, ESI, DS, ES, FS, GS destroyed
Note: in protected mode, calling the protected-mode VCPI entry point with
    AX = DE0Ch
    DS = segment selector mapping entire linear address space obtained
      via AX=DE01h
    SS:ESP in first megabyte of linear memory
    STACK:QWORD  return address from FAR call to 32-bit segment
    DWORD  EIP
    DWORD  CS
    DWORD  reserved for EFLAGS
    DWORD  ESP
    DWORD  SS
    DWORD  ES
    DWORD  DS
    DWORD  FS
    DWORD  GS
    and interrupts disabled, will switch to virtual86 mode with
      interrupts disabled, all segment registers loaded, and EAX
      destroyed.
SeeAlso: AH=3Fh,INT 15/AH=89h,INT D4/AH=10h

Format of system register values for switch to protected mode:
Offset  Size  Description (Table 03665)
 00h  DWORD value for CR3
 04h  DWORD linear address in first megabyte of value for GDTR
```

```
 08h  DWORD linear address in first megabyte of value for IDTR
 0Ch  WORD  value for LDTR
 0Eh  WORD  value for TR
 10h  PWORD CS:EIP of protected mode entry-point
--------m-67DE0F-----------------------------
INT 67 - Netroom3 - ???
  AX = DE0Fh
  ???
Return: ???
Note: called by Netroom's DPMI.EXE
--------s-67DEE1BX0C55-----------------------
INT 67 - "SB Live!" Sound Blaster 16 Emulation Driver - INSTALLATION CHECK
  AX = DEE1h
  BX = 0C55h
Return: BL = AAh if installed
--------m-67EF-------------------------------
INT 67 - RM386 v6.00+ - EXECUTE XMS FUNCTION
  AH = EFh
  AL = function (00h-12h,80h-8Fh)
  other register as appropriate for XMS function
Return: varies by function (see INT 2F/AX=4310h"XMS")
Note: these functions appear to be equivalent to the XMS functions with the
    same numbers
SeeAlso: AX=5DFFh"RM386",INT 2F/AX=4310h"XMS"
--------m-67FFA5-----------------------------
INT 67 - Microsoft EMM386.EXE v4.20+ - INSTALLATION CHECK
  AX = FFA5h
Return: AX = 845Ah/84A5h if loaded
      BX:CX -> API entry point (see #03666)
Notes:  this call is available even if EMM386 is not providing EMS
  the returned AX is 845Ah inside of MSWindows, 84A5h under bare DOS
  if no other program has hooked INT 67, an alternate installation
    check is to test for the string
    "MICROSOFT EXPANDED MEMORY MANAGER 386" at offset 14h in the INT 67
    handler's segment; the word immediately preceding this string
    contains the offset of the API entry point
SeeAlso: AH=3Fh,AX=FFA5h/BX=4345h,INT 21/AX=4402h"EMM386.EXE"

(Table 03666)
Call EMM386.EXE API entry point with:
  AH = 00h get memory manager's status
```

```
      Return: AH = status
      bit 0: not active (OFF)
      bit 1: in "Auto" mode
  AH = 01h set memory manager's state
      AL = new state (00h ON, 01h OFF, 02h AUTO)
  AH = 02h Weitek coprocessor support
      AL = subfunction
    00h get Weitek support state
        Return: AL = status
        bit 0: Weitek coprocessor is present
        bit 1: Weitek support is enabled
    01h turn on Weitek support
    02h turn off Weitek support
     --- v4.20-4.41 only ---
  AH = 03h Windows support???
      AL = subfunction (00h, 01h)
  AH = 04h print copyright notice to standard output
     (using INT 21/AH=09h)
  AH = 05h print available report
     (the one shown when running EMM386 from the DOS prompt)
SeeAlso: #01513 at INT 21/AX=4402h/SF=02h,#02617 at INT 2F/AX=12FFh/BX=0106h
--------m-67FFA5BX4345-----------------------
INT 67 U - Compaq CEMM v5.10+ - PRIVATE API
  AX = FFA5h
  BX = 4345h ("CE")
  DX = subfunction
      0000h unshadow video ROM???
      0001h shadow video ROM???
      0002h map pages
    CX = number of pages (00h=one)
    ESI = linear address of first page to map into address space
    EDI = linear starting address at which pages are to be visible
      0003h get ???
    Return: DX = ??? (0-2)
      0004h BUG: crashes system due to fencepost error
Return: AH = 84h
  AL = status (84h = error, FFh = success)
Note: if BX <> 4345h or DX > 0004h on entry, CEMM behaves identically to
    Microsoft's EMM386 (see AX=FFA5h"EMM386")
SeeAlso: AX=FFA5h"EMM386",#01513 at INT 21/AX=4402h/SF=02h,#03666
--------I-68---------------------------------
```

```
INT 68 - Sangoma CCPOP 3270 resident module
SeeAlso: INT 67"Sangoma",INT 92"Sangoma"
--------N-68---------------------------------
INT 68 - Novell NetWare LU6.2
InstallCheck: test for the signature string "APPC/PC" nine bytes before the
    interrupt handler
SeeAlso: AH=01h/SF=1B00h,AH=FAh
--------h-68---------------------------------
INT 68 C - HP Vectra AT - IRQ16 - 8041 SERVICE REQUEST
SeeAlso: INT 08"IRQ0",INT 69"HP Vectra",INT 6F/AH=00h"HP"
--------N-6801--SF1B00-----------------------
INT 68 - APPC/PC - NETWORK DEVICE CONTROL - DISPLAY
  AH = 01h subfn 1B00h
  DS:DX -> control block (see #03667)
Return: control block updated
SeeAlso: AH=01h/SF=2000h,AH=01h/SF=2B00h,INT 68"Novell"

Format of APPC/PC "DISPLAY" control block:
Offset  Size  Description (Table 03667)
 00h 12 BYTEs reserved
 0Ch  WORD  1B00h (verb "DISPLAY")
 0Eh  6 BYTEs 00h
 14h  DWORD (big-endian) return code (see #03668)
 18h  WORD  00h
 1Ah  8 BYTEs (big-endian) logical unit ID
 22h  8 BYTEs (big-endian) partner logical unit name
 2Ah  8 BYTEs (big-endian) mode name
 32h  BYTE  logical unit session limit
 33h  BYTE  partner logical unit session limit
 34h  BYTE  node maximum negotiable session limit
 35h  BYTE  current session limit
 36h  BYTE  minimum negotiated winner limit
 37h  BYTE  maximum negotiated loser limit
 38h  BYTE  active session count
 39h  BYTE  active CONWINNER session count
 3Ah  BYTE  active CONLOSER session count
 3Bh  BYTE  session termination count
 3Ch  BYTE  bit 7: SESSION_TERMINATION_TARGET_DRAIN
    bit 6: SESSION_TERMINATION_SOURCE_DRAIN

(Table 03668)
```

```
Values for APPC/PC return code:
 0000h  successful
 0001h  BAD_TP_ID
 0002h  BAD_CONV_ID
 0003h  bad logical unit ID
 0008h  no physical unit attached
 0110h  bad state
 01B1h  BAD_PART_LUNAME
 01B2h  bad mode name
 0201h  physical unit already active
 0211h  logical unit already active
 0212h  BAD_PART_SESS
 0213h  BAD_RU_SIZES
 0214h  BAD_MODE_SESS
 0216h  BAD_PACING_CNT
 0219h  EXTREME_RUS
 021Ah  SNASVCMG_1
 0223h  SSCP_CONNECTED_LU
 0230h  invalid change
 0243h  too many TPs
 0272h  adapter close failure
 0281h  GET_ALLOC_BAD_TYPE
 0282h  unsuccessful
 0283h  DLC failure
 0284h  unrecognized DLC
 0286h  duplicate DLC
 0301h  SSCP_PU_SESSION_NOT_ACTIVE
 0302h  data exceeds RU size
 0401h  invalid direction
 0402h  invalid type
 0403h  segment overlap
 0404h  invalid first character
 0405h  table error
 0406h  conversion error
 F0010000h  APPC disabled
 F0020000h  APPC busy
 F0030000h  APPC abended
 F0040000h  incomplete
--------N-6801--SF2000-----------------------
INT 68 - APPC/PC - NETWORK DEVICE CONTROL - ATTACH PHYSICAL UNIT
  AH = 01h subfn 2000h
```

```
  DS:DX -> control block (see #03669)
Return: control block updated
SeeAlso: AH=01h/SF=2100h,AH=01h/SF=2B00h

Format of APPC/PC "Attach Physical Unit" control block:
Offset  Size  Description (Table 03669)
 00h 12 BYTEs reserved
 0Ch  WORD  2000h (verb "Attach Physical Unit")
 0Eh  6 BYTEs 00h
 14h  DWORD (big-endian) return code (see #03668)
 18h  WORD  00h
 1Ah  BYTE  version
 1Bh  BYTE  release
 1Ch  8 BYTEs (big-endian) net name
 24h  8 BYTEs (big-endian) physical unit name
 2Ch  8 BYTEs 00h
 34h  DWORD pointer to SYSTEM_LOG_EXIT routine, FFFFFFFFh=don't log errors
    (see also AH=01h/SF=2100h)
 38h  DWORD 00h
 3Ch  BYTE  00h RETURN_CONTROL: COMPLETE
    01h RETURN_CONTROL: INCOMPLETE
--------N-6801--SF2100-----------------------
INT 68 - APPC/PC - NETWORK DEVICE CONTROL - ATTACH LOGICAL UNIT
  AH = 01h subfn 2100h
  DS:DX -> control block (see #03670)
Return: control block updated
SeeAlso: #03673,#03674,#03676,AH=01h/SF=2000h,AH=01h/SF=2200h,AH=01h/SF=2B00h

Format of APPC/PC "Attach Logical Unit" control block:
Offset  Size  Description (Table 03670)
 00h 12 BYTEs reserved
 0Ch  WORD  2100h (verb "Attach Logical Unit")
 0Eh  6 BYTEs 00h
 14h  DWORD (big-endian) return code (see #03668)
 18h  WORD  70  offset to partner logical unit record
 1Ah  8 BYTEs (big-endian) logical unit name
 22h  8 BYTEs (big-endian) logical unit ID
 2Ah  BYTE  logical unit local address
 2Bh  BYTE  logical unit session limit
 2Ch  DWORD pointer to CREATE_TP_EXIT routine,
    FFFFFFFFh = reject incoming ALLOCATEs
```

```
    00000000h = queue ALLOCATEs
 30h  DWORD 00h
 34h  DWORD pointer to SYSTEM_LOG_EXIT routine, FFFFFFFFh=don't log errors
 38h  DWORD 00h
 3Ch  BYTE  maximum TPs
 3Dh  BYTE  queue depth
 3Eh  DWORD pointer to LU_LU_PASSWORD_EXIT routine, FFFFFFFFh=no pswd exit
 42h  DWORD 00h
 46h  WORD  total length of partner records
 48h  var array of partner logical unit records (see #03671)
SeeAlso: #03673,#03674,#03676

Format of APPC/PC partner logical unit record:
Offset  Size  Description (Table 03671)
 00h  WORD  length of this partner logical unit record
 02h  WORD  42  offset to mode records
 04h  8 BYTEs (big-endian) partner logical unit name
 0Ch  BYTE  partner logical unit security capabilities
    bit 7: already verified
    bit 6: conversation level security
    bit 5: session level security
 0Dh  BYTE  partner logical unit session limit
 0Eh  WORD  partner logical unit maximum MC_SEND_LL
 10h  8 BYTEs (big-endian) partner logical unit DLC name
 18h  BYTE  partner logical unit adapter number
 19h 17 BYTEs (counted string) partner logical unit adapter address
 2Ah  WORD  total length of mode records
 2Ch 16N BYTEs  array of mode records (see #03672)

Format of mode record:
Offset  Size  Description (Table 03672)
 00h  WORD  16  length of this mode record
 02h  8 BYTEs (big-endian) mode name
 0Ah  WORD  RU_SIZE high bound
 0Ch  WORD  RU_SIZE low bound
 0Eh  BYTE  mode maximum negotiable session limit
 0Fh  BYTE  pacing size for receive

Routines defined by LU_LU_PASSWORD_EXIT, CREATE_TP_EXIT, and SYSTEM_LOG_EXIT
pointers are called by pushing the DWORD pointer to the verb on the stack and
then performing a FAR call.
```

```
Format of ACCESS_LU_LU_PW verb:
Offset  Size  Description (Table 03673)
 00h 12 BYTEs reserved
 0Ch  WORD  1900h (verb "ACCESS_LU_LU_PW")
 0Eh  8 BYTEs (big-endian) logical unit ID
 16h  8 BYTEs (big-endian) logical unit name
 1Eh  8 BYTEs (big-endian) partner logical unit name
 26h 17 BYTEs (counted string) partner fully qualified logical unit name
 37h  BYTE  password available (0=no, 1=yes)
 38h  8 BYTEs password
SeeAlso: #03670,#03674,#03676

Format of CREATE_TP verb:
Offset  Size  Description (Table 03674)
 00h 12 BYTEs reserved
 0Ch  WORD  2300h (verb "CREATE_TP")
 0Eh  6 BYTEs 00h
 14h  DWORD (big-endian) sense code (see #03675)
 18h  8 BYTEs (big-endian) TP ID
 20h  8 BYTEs (big-endian) logical unit ID
 28h  DWORD (big-endian) conversation ID
 2Ch  BYTE  0 basic conversation, 1 mapped conversation
 2Dh  BYTE  0 no sync level, 1 confirm
 2Eh  BYTE  reserved
 2Fh 65 BYTEs (counted string) transaction program name
 70h  6 BYTEs 00h
 76h  WORD  length of ERROR_LOG_DATA to return
 78h  DWORD pointer to ERROR_LOG_DATA buffer
 7Ch  8 BYTEs (big-endian) partner logical unit name
 84h 18 BYTEs (counted string) partner fully qualified logical unit name
 96h  8 BYTEs (big-endian) mode name
 9Eh 12 BYTEs 00h
 AAh 11 BYTEs (counted string) password
 B5h 11 BYTEs (counted string) user ID
 C0h  BYTE  0 verification should be performed
    1 already verified
SeeAlso: #03673,#03676

(Table 03675)
Values for APPC/PC sense code:
```

```
 00000000h  Ok
 080F6051h  SECURITY_NOT_VALID
 084B6031h  TP_NOT_AVAIL_RETRY
 084C0000h  TP_NOT_AVAIL_NO_RETRY
 10086021h  TP_NAME_NOT_RECOGNIZED
 10086034h  CONVERSATION_TYPE_MISMATCH
 10086041h  SYNC_LEVEL_NOT_SUPPORTED

Format of SYSLOG verb:
Offset  Size  Description (Table 03676)
 00h 12 BYTEs reserved
 0Ch  WORD  2600h (verb "SYSLOG")
 0Eh 10 BYTEs 00h
 18h  WORD  (big-endian) type
 1Ah  DWORD (big-endian) subtype
 1Eh  DWORD pointer to ADDITIONAL_INFO
 22h  DWORD (big-endian) conversation ID
 26h  8 BYTEs (big-endian) TP ID
 2Eh  8 BYTEs (big-endian) physical unit or logical unit name
 36h  WORD  length of data
 38h  DWORD pointer to data
 3Ch  BYTE  00h
SeeAlso: #03673,#03674
--------N-6801--SF2200-----------------------
INT 68 - APPC/PC - NETWORK DEVICE CONTROL - DETACH LOGICAL UNIT
  AH = 01h subfn 2200h
  DS:DX -> control block (see #03677)
Return: control block updated
SeeAlso: AH=01h/SF=2000h,AH=01h/SF=2100h,AH=01h/SF=2700h

Format of APPC/PC "Detach Logical Unit" control block:
Offset  Size  Description (Table 03677)
 00h 12 BYTEs reserved
 0Ch  WORD  2200h (verb "Detach Logical Unit")
 0Eh  6 BYTEs 00h
 14h  DWORD (big-endian) return code (see #03668)
 18h  8 BYTEs (big-endian) logical unit ID
 20h  BYTE  00h
--------N-6801--SF2700-----------------------
INT 68 - APPC/PC - NETWORK DEVICE CONTROL - DETACH PHYSICAL UNIT
  AH = 01h subfn 2700h
```

```
  DS:DX -> control block (see #03678)
Return: control block updated
SeeAlso: AH=01h/SF=2000h,AH=01h/SF=2100h,AH=01h/SF=2200h

Format of APPC/PC "Detach Physical Unit" control block:
Offset  Size  Description (Table 03678)
 00h 12 BYTEs reserved
 0Ch  WORD  2700h (verb "Detach Physical Unit")
 0Eh  6 BYTEs 00h
 14h  DWORD (big-endian) return code (see #03668)
 18h  BYTE  00h  type: hard
    01h  type: soft
--------N-6801--SF2B00----------------------
INT 68 - APPC/PC - NETWORK DEVICE CONTROL - ACTIVATE DLC
  AH = 01h subfn 2B00h
  DS:DX -> control block (see #03679)
Return: control block updated
SeeAlso: AH=01h/SF=1B00h,AH=01h/SF=2000h

Format of APPC/PC "Activate DLC" control block:
Offset  Size  Description (Table 03679)
 00h 12 BYTEs reserved
 0Ch  WORD  2B00h (verb "Activate DLC")
 0Eh  6 BYTEs 00h
 14h  DWORD (big-endian) return code (see #03668)
 18h  8 BYTEs (big-endian) DLC name
 20h  BYTE  adapter number
--------N-6802--SF0100----------------------
INT 68 - APPC/PC - CONNECTION CONTROL - ALLOCATE
  AH = 02h subfn 0100h
  DS:DX -> control block (see #03680)
Return: control block updated
SeeAlso: AH=02h/SF=0500h

Format of APPC/PC "Allocate" control block:
Offset  Size  Description (Table 03680)
 00h 12 BYTEs reserved
 0Ch  WORD  0100h (verb "Allocate" or "MC_Allocate")
 0Eh  BYTE  1 if MC_ (mapped conversation) form of verb
    0 if basic verb
 0Fh  5 BYTEs reserved (0)
```

```
 14h  WORD  (big-endian) primary return code (see #03681)
 16h  DWORD (big-endian) error code (see #03682)
 1Ah  8 BYTEs (big-endian) TP_ID
 22h  DWORD (big-endian) conversation ID
 26h  BYTE  (MC_Allocate only) conversation type
    0 basic conversation
    1 mapped conversation
 27h  BYTE  SYNC_LEVEL (00h none, 01h confirm)
 28h  WORD  0000h
 2Ah  BYTE  RETURN_CONTROL
    00h when session allocated
    01h immediate
    02h when session free
 2Bh  8 BYTEs 00h
 33h  8 BYTEs (big-endian) partner logical unit name
 3Bh  8 BYTEs (big-endian) mode name
 43h 65 BYTEs (counted string) TP name
 84h  BYTE  security (00h none, 01h same, 02h pgm)
 85h 11 BYTEs 00h
 90h 11 BYTEs (counted string) password
 9Bh 11 BYTEs (counted string) user ID
 A6h  WORD  PIP_DATA length
 A8h  DWORD pointer to PIP_DATA

(Table 03681)
Values for APPC/PC primary return code:
 0000h  successful
 0001h  parameter check
 0002h  state check
 0003h  allocation error
 0005h  deallocate abended
 0006h  deallocate abended program
 0007h  deallocate abended SVC
 0008h  deallocate abended timer
 0009h  deallocate normal return
 000Ah  data posting blocked
 000Bh  posting not active
 000Ch  PROG_ERROR_NO_TRUNC
 000Dh  PROG_ERROR_TRUNC
 000Eh  PROG_ERROR_PURGING
 000Fh  CONV_FAILURE_RETRY
```

```
 0010h  CONV_FAILURE_NO_RETRY
 0011h  SVC_ERROR_NO_TRUNC
 0012h  SVC_ERROR_TRUNC
 0013h  SVC_ERROR_PURGING
 0014h  unsuccessful
 0018h  CNOS partner logical unit reject
 0019h  conversation type mixed
 F001h  APPC disabled
 F002h  APPC busy
 F003h  APPC abended
 F004h  incomplete

(Table 03682)
Values for APPC/PC error code:
 0001h  bad TP ID
 0002h  bad conversation ID
 0004h  allocation error, no retry
 0005h  allocation error, retry
 0006h  data area crosses segment boundary
 0010h  bad TPN length
 0011h  bad CONV length
 0012h  bad SYNC level
 0013h  bad security selection
 0014h  bad return control
 0015h  SEC_TOKENS too big
 0016h  PIP_LEN incorrect
 0017h  no use of SNASVCMG
 0018h  unknown partner mode
 0031h  confirm: SYNC_NONE
 0032h  confirm: bad state
 0033h  confirm: NOT_LL_BDY
 0041h  confirmed: bad state
 0051h  deallocate: bad type
 0052h  deallocate: flush bad state
 0053h  deallocate: confirm bad state
 0055h  deallocate: NOT_LL_BDY
 0057h  deallocate: log LL_WRONG
 0061h  flush: not send state
 0091h  post on receipt: invalid length
 0092h  post on receipt: not in receive state
 0093h  post on receipt: bad fill
```

```
 00A1h  prepare to receive:invalid type
 00A2h  prepare to receive: unfinished LL
 00A3h  prepare to receive: not in send state
 00B1h  receive and wait: bad state
 00B2h  receive and wait: NOT_LL_BDY
 00B5h  receive and wait: bad fill
 00C1h  receive immediate: not in receive state
 00C4h  receive immediate: bad fill
 00E1h  request to send: not in receive state
 00F1h  send data: bad LL
 00F2h  send data: not in send state
 0102h  send error: log LL wrong
 0103h  send error: bad type
 0121h  test: invalid type
 0122h  test: not in receive state
--------N-6802--SF0300-----------------------
INT 68 - APPC/PC - CONNECTION CONTROL - CONFIRM
  AH = 02h subfn 0300h
  DS:DX -> control block (see #03683)
Return: control block updated
SeeAlso: AH=02h/SF=0400h

Format of APPC/PC "Confirm" control block:
Offset  Size  Description (Table 03683)
 00h 12 BYTEs reserved
 0Ch  WORD  0300h (verb "Confirm" or "MC_Confirm")
 0Eh  BYTE  1 if MC_ (mapped conversation) form of verb
    0 if basic verb
 0Fh  5 BYTEs reserved (0)
 14h  WORD  (big-endian) primary return code (see #03681)
 16h  DWORD (big-endian) error code (see #03682)
 1Ah  8 BYTEs (big-endian) TP_ID
 22h  DWORD (big-endian) conversation ID
 26h  BYTE  request to send received (0=no, 1=yes)
--------N-6802--SF0400-----------------------
INT 68 - APPC/PC - CONNECTION CONTROL - CONFIRMED
  AH = 02h subfn 0400h
  DS:DX -> control block (see #03684)
Return: control block updated
SeeAlso: AH=02h/SF=0300h
```

```
Format of APPC/PC "Confirmed" control block:
Offset  Size  Description (Table 03684)
 00h 12 BYTEs reserved
 0Ch  WORD  0400h (verb "Confirmed" or "MC_Confirmed")
 0Eh  BYTE  1 if MC_ (mapped conversation) form of verb
    0 if basic verb
 0Fh  5 BYTEs reserved (0)
 14h  WORD  (big-endian) primary return code (see #03681)
 16h  DWORD (big-endian) error code (see #03682)
 1Ah  8 BYTEs (big-endian) TP_ID
 22h  DWORD (big-endian) conversation ID
--------N-6802--SF0500-----------------------
INT 68 - APPC/PC - CONNECTION CONTROL - DEALLOCATE
  AH = 02h subfn 0500h
  DS:DX -> control block (see #03685)
Return: control block updated
SeeAlso: AH=02h/SF=0100h,AH=02h/SF=0300h

Format of APPC/PC "Deallocate" control block:
Offset  Size  Description (Table 03685)
 00h 12 BYTEs reserved
 0Ch  WORD  0500h (verb "Deallocate" or "MC_Deallocate")
 0Eh  BYTE  1 if MC_ (mapped conversation) form of verb
    0 if basic verb
 0Fh  5 BYTEs reserved (0)
 14h  WORD  (big-endian) primary return code (see #03681)
 16h  DWORD (big-endian) error code (see #03682)
 1Ah  8 BYTEs (big-endian) TP_ID
 22h  DWORD (big-endian) conversation ID
 26h  BYTE  00h
 27h  BYTE  type
    00h SYNC_LEVEL
    01h FLUSH
    02h ABEND_PROC
    03h ABEND_SVC
    04h ABEND_TIMER
    05h ABEND
 28h  WORD  (MC_Deallocate only) length of error log data
 2Ah  DWORD (MC_Deallocate only) pointer to error log data
--------N-6802--SF0600-----------------------
INT 68 - APPC/PC - CONNECTION CONTROL - FLUSH
```

```
  AH = 02h subfn 0600h
  DS:DX -> control block (see #03686)
Return: control block updated
SeeAlso: AH=02h/SF=0300h

Format of APPC/PC "Flush" control block:
Offset  Size  Description (Table 03686)
 00h 12 BYTEs reserved
 0Ch  WORD  0600h (verb "Flush" or "MC_Flush")
 0Eh  BYTE  1 if MC_ (mapped conversation) form of verb
    0 if basic verb
 0Fh  5 BYTEs reserved (0)
 14h  WORD  (big-endian) primary return code (see #03681)
 16h  DWORD (big-endian) error code (see #03682)
 1Ah  8 BYTEs (big-endian) TP_ID
 22h  DWORD (big-endian) conversation ID
--------N-6802--SF0700----------------------
INT 68 - APPC/PC - CONNECTION CONTROL - GET ATTRIBUTES
  AH = 02h subfn 0700h
  DS:DX -> control block (see #03687)
Return: control block updated
SeeAlso: AH=02h/SF=0300h

Format of APPC/PC "Get_Attributes" control block:
Offset  Size  Description (Table 03687)
 00h 12 BYTEs reserved
 0Ch  WORD  0700h (verb "Get_Attributes" or "MC_Get_Attributes")
 0Eh  BYTE  1 if MC_ (mapped conversation) form of verb
    0 if basic verb
 0Fh  5 BYTEs reserved (0)
 14h  WORD  (big-endian) primary return code (see #03681)
 16h  DWORD (big-endian) error code (see #03682)
 1Ah  8 BYTEs (big-endian) TP_ID
 22h  DWORD (big-endian) conversation ID
 26h  8 BYTEs (big-endian) logical unit ID
 2Eh  BYTE  00h
 2Fh  BYTE  SYNC_LEVEL (0=none, 1=confirm)
 30h  8 BYTEs (big-endian) mode name
 38h  8 BYTEs (big-endian) own net name
 40h  8 BYTEs (big-endian) own logical unit name
 48h  8 BYTEs (big-endian) partner logical unit name
```

```
 50h 18 BYTEs (counted string) partner's fully qualified logical unit name
 62h  BYTE  00h
 63h 11 BYTEs (counted string) user ID
--------N-6802--SF0800-----------------------
INT 68 - APPC/PC - CONNECTION CONTROL - GET CONVERSATION TYPE
  AH = 02h subfn 0800h
  DS:DX -> control block (see #03688)
Return: control block updated
SeeAlso: AH=02h/SF=0300h

Format of APPC/PC "Get_Type" control block:
Offset  Size  Description (Table 03688)
 00h 12 BYTEs reserved
 0Ch  WORD  0800h (verb "Get_Type")
 0Eh  BYTE  1 if MC_ (mapped conversation) form of verb
    0 if basic verb
 0Fh  5 BYTEs reserved (0)
 14h  WORD  (big-endian) primary return code (see #03681)
 16h  DWORD (big-endian) error code (see #03682)
 1Ah  8 BYTEs (big-endian) TP_ID
 22h  DWORD (big-endian) conversation ID
 26h  BYTE  (ret) type (0=basic conversation, 1=mapped conversation)
--------N-6802--SF0900-----------------------
INT 68 - APPC/PC - CONNECTION CONTROL - POST ON RECEIPT
  AH = 02h subfn 0900h
  DS:DX -> control block (see #03689)
Return: control block updated
SeeAlso: AH=02h/SF=0A00h

Format of APPC/PC "Post_on_Receipt" control block:
Offset  Size  Description (Table 03689)
 00h 12 BYTEs reserved
 0Ch  WORD  0900h (verb "Post_on_Receipt")
 0Eh  BYTE  1 if MC_ (mapped conversation) form of verb
    0 if basic verb
 0Fh  5 BYTEs reserved (0)
 14h  WORD  (big-endian) primary return code (see #03681)
 16h  DWORD (big-endian) error code (see #03682)
 1Ah  8 BYTEs (big-endian) TP_ID
 22h  DWORD (big-endian) conversation ID
 26h  WORD  maximum length
```

```
 28h  BYTE  fill (0=buffer, 1=LL)
--------N-6802--SF0A00-----------------------
INT 68 - APPC/PC - CONNECTION CONTROL - PREPARE TO RECEIVE
  AH = 02h subfn 0A00h
  DS:DX -> control block (see #03690)
Return: control block updated
SeeAlso: AH=02h/SF=0900h,AH=02h/SF=0B00h

Format of APPC/PC "Prepare_to_Receive" control block:
Offset  Size  Description (Table 03690)
 00h 12 BYTEs reserved
 0Ch  WORD  0A00h (verb "Prepare_to_Receive" or "MC_Prepare_to_Receive")
 0Eh  BYTE  1 if MC_ (mapped conversation) form of verb
    0 if basic verb
 0Fh  5 BYTEs reserved (0)
 14h  WORD  (big-endian) primary return code (see #03681)
 16h  DWORD (big-endian) error code (see #03682)
 1Ah  8 BYTEs (big-endian) TP_ID
 22h  DWORD (big-endian) conversation ID
 26h  BYTE  type (0=SYNC_LEVEL, 1=FLUSH)
 27h  BYTE  locks (0=short, 1=long)
--------N-6802--SF0B00-----------------------
INT 68 - APPC/PC - CONNECTION CONTROL - RECEIVE AND WAIT
  AH = 02h subfn 0B00h
  DS:DX -> control block (see #03691)
Return: control block updated
SeeAlso: AH=02h/SF=0C00h,AH=02h/SF=0F00h

Format of APPC/PC "Receive_and_Wait" control block:
Offset  Size  Description (Table 03691)
 00h 12 BYTEs reserved
 0Ch  WORD  0B00h (verb "Receive_and_Wait" or "MC_Receive_and_Wait")
 0Eh  BYTE  1 if MC_ (mapped conversation) form of verb
    0 if basic verb
 0Fh  5 BYTEs reserved (0)
 14h  WORD  (big-endian) primary return code (see #03681)
 16h  DWORD (big-endian) error code (see #03682)
 1Ah  8 BYTEs (big-endian) TP_ID
 22h  DWORD (big-endian) conversation ID
 26h  BYTE  type of information received (see #03692)
 27h  BYTE  (MC_Receive_and_Wait only) fill (0=buffer, 1=LL)
```

```
 28h  BYTE  Request_to_Send_Received (0=no, 1=yes)
 29h  WORD  maximum length
 2Bh  WORD  data length
 2Dh  DWORD pointer to data

(Table 03692)
Values for type of information received:
 00h  data
 01h  data complete
 02h  data incomplete
 03h  confirm
 04h  confirm send
 05h  confirm deallocate
 06h  send
--------N-6802--SF0C00----------------------
INT 68 - APPC/PC - CONNECTION CONTROL - RECEIVE IMMEDIATE
  AH = 02h subfn 0C00h
  DS:DX -> control block (see #03693)
Return: control block updated
SeeAlso: AH=02h/SF=0B00h,AH=02h/SF=0F00h

Format of APPC/PC "Receive_Immediate" control block:
Offset  Size  Description (Table 03693)
 00h 12 BYTEs reserved
 0Ch  WORD  0C00h (verb "Receive_Immediate" or "MC_Receive_Immediate")
 0Eh  BYTE  1 if MC_ (mapped conversation) form of verb
    0 if basic verb
 0Fh  5 BYTEs reserved (0)
 14h  WORD  (big-endian) primary return code (see #03681)
 16h  DWORD (big-endian) error code (see #03682)
 1Ah  8 BYTEs (big-endian) TP_ID
 22h  DWORD (big-endian) conversation ID
 26h  BYTE  type of information received (see #03692)
 27h  BYTE  (MC_Receive_Immediate only) fill (0=buffer, 1=LL)
 28h  BYTE  Request_to_Send_Received (0=no, 1=yes)
 29h  WORD  maximum length
 2Bh  WORD  data length
 2Dh  DWORD pointer to data
--------N-6802--SF0E00----------------------
INT 68 - APPC/PC - CONNECTION CONTROL - REQUEST TO SEND
  AH = 02h subfn 0E00h
```

```
  DS:DX -> control block (see #03694)
Return: control block updated
SeeAlso: AH=02h/SF=0F00h,AH=02h/SF=1000h

Format of APPC/PC "Request_to_Send" control block:
Offset  Size  Description (Table 03694)
 00h 12 BYTEs reserved
 0Ch  WORD  0E00h (verb "Request_to_Send" or "MC_Request_to_Send")
 0Eh  BYTE  1 if MC_ (mapped conversation) form of verb
    0 if basic verb
 0Fh  5 BYTEs reserved (0)
 14h  WORD  (big-endian) primary return code (see #03681)
 16h  DWORD (big-endian) error code (see #03682)
 1Ah  8 BYTEs (big-endian) TP_ID
 22h  DWORD (big-endian) conversation ID
--------N-6802--SF0F00----------------------
INT 68 - APPC/PC - CONNECTION CONTROL - SEND DATA
  AH = 02h subfn 0F00h
  DS:DX -> control block (see #03695)
Return: control block updated
SeeAlso: AH=02h/SF=0E00h,AH=02h/SF=1000h

Format of APPC/PC "Send_Data" control block:
Offset  Size  Description (Table 03695)
 00h 12 BYTEs reserved
 0Ch  WORD  0F00h (verb "Send_Data" or "MC_Send_Data")
 0Eh  BYTE  1 if MC_ (mapped conversation) form of verb
    0 if basic verb
 0Fh  5 BYTEs reserved (0)
 14h  WORD  (big-endian) primary return code (see #03681)
 16h  DWORD (big-endian) error code (see #03682)
 1Ah  8 BYTEs (big-endian) TP_ID
 22h  DWORD (big-endian) conversation ID
 26h  BYTE  request to send received (0=no, 1=yes)
 27h  BYTE  00h
 28h  WORD  data length
 2Ah  DWORD pointer to data
--------N-6802--SF1000----------------------
INT 68 - APPC/PC - CONNECTION CONTROL - SEND ERROR
  AH = 02h subfn 1000h
  DS:DX -> control block (see #03696)
```

```
Return: control block updated
SeeAlso: AH=02h/SF=0F00h

Format of APPC/PC "Send_Error" control block:
Offset  Size  Description (Table 03696)
 00h 12 BYTEs reserved
 0Ch  WORD  1000h (verb "Send_Error" or "MC_Send_Error")
 0Eh  BYTE  1 if MC_ (mapped conversation) form of verb
    0 if basic verb
 0Fh  5 BYTEs reserved (0)
 14h  WORD  (big-endian) primary return code (see #03681)
 16h  DWORD (big-endian) error code (see #03682)
 1Ah  8 BYTEs (big-endian) TP_ID
 22h  DWORD (big-endian) conversation ID
 26h  BYTE  request to send received (0=no, 1=yes)
 27h  BYTE  type (0=program, 1=SVC)
 28h  DWORD 00h
 2Ch  WORD  (MC_Send_Error only) LOG_DATA length
 2Eh  DWORD (MC_Send_Error only) pointer to LOG_DATA
--------N-6802--SF1200-----------------------
INT 68 - APPC/PC - CONNECTION CONTROL - TEST
  AH = 02h subfn 1200h
  DS:DX -> control block (see #03697)
Return: control block updated
SeeAlso: AH=02h/SF=1300h

Format of APPC/PC "Test" control block:
Offset  Size  Description (Table 03697)
 00h 12 BYTEs reserved
 0Ch  WORD  1200h (verb "Test" or "MC_Test")
 0Eh  BYTE  1 if MC_ (mapped conversation) form of verb
    0 if basic verb
 0Fh  5 BYTEs reserved (0)
 14h  WORD  (big-endian) primary return code (see #03681)
 16h  DWORD (big-endian) error code (see #03682)
 1Ah  8 BYTEs (big-endian) TP_ID
 22h  DWORD (big-endian) conversation ID
 26h  BYTE  (MC_Test only) test (0=posted, 1=request_to_send received)
Note: error code has different interpretations for:
  0 posted data
  1 posted not data (primary return code = 0)
```

```
  1 bad TP_ID (primary return code = 1)
--------N-6802--SF1300-----------------------
INT 68 - APPC/PC - CONNECTION CONTROL - WAIT
  AH = 02h subfn 1300h
  DS:DX -> control block (see #03698)
Return: control block updated
SeeAlso: AH=02h/SF=1200h

Format of APPC/PC "Wait" control block:
Offset  Size  Description (Table 03698)
 00h 12 BYTEs reserved
 0Ch  WORD  1300h (verb "Wait")
 0Eh  BYTE  1 if MC_ (mapped conversation) form of verb
    0 if basic verb
 0Fh  5 BYTEs reserved (0)
 14h  WORD  (big-endian) primary return code (see #03681)
 16h  DWORD (big-endian) error code (see #03682,#03697)
 1Ah  8 BYTEs (big-endian) TP_ID
 22h  DWORD (big-endian) conversation ID
 26h  BYTE  number of conversations to wait on
    Note: error codes have interpretations as for AH=02h/SF=1200h
--------N-6803--SF2400-----------------------
INT 68 - APPC/PC - TP STARTED
  AH = 03h subfn 2400h
  DS:DX -> control block (see #03699)
Return: control block updated

Format of APPC/PC "TP Started" control block:
Offset  Size  Description (Table 03699)
 00h 12 BYTEs reserved
 0Ch  WORD  2400h (verb "TP Started")
 0Eh  6 BYTEs 00h
 14h  DWORD (big-endian) return code (see #03668)
 18h  WORD  00h
 1Ah  8 BYTEs (big-endian) logical unit ID
 22h  8 BYTEs (big-endian) TP ID
--------N-6803--SF2800-----------------------
INT 68 - APPC/PC - GET ALLOCATE
  AH = 03h subfn 2800h
  DS:DX -> control block (see #03700)
Return: control block updated
```

```
Format of APPC/PC "Get ALLOCATE" control block:
Offset  Size  Description (Table 03700)
 00h 12 BYTEs reserved
 0Ch  WORD  2800h (verb "Get ALLOCATE")
 0Eh  6 BYTEs 00h
 14h  DWORD (big-endian) return code (see #03668)
 18h  WORD  00h
 1Ah  8 BYTEs (big-endian) logical unit ID
 22h  BYTE  type (00h dequeue, 01h test)
 23h  DWORD pointer to CREATE_TP record
--------N-6803--SF2A00-----------------------
INT 68 - APPC/PC - CHANGE LOGICAL UNIT
  AH = 03h subfn 2A00h
  DS:DX -> control block (see #03701)
Return: control block updated

Format of APPC/PC "Change Logical Unit" control block:
Offset  Size  Description (Table 03701)
 00h 12 BYTEs reserved
 0Ch  WORD  2A00h (verb "Change Logical Unit")
 0Eh  6 BYTEs 00h
 14h  DWORD (big-endian) return code (see #03668)
 18h  WORD  00h
 1Ah  8 BYTEs (big-endian) logical unit ID
 22h  DWORD pointer to CREATE_TP_EXIT routine
    00000000h queue ALLOCATEs
    FFFFFFFFh reject incoming ALLOCATEs
 26h  DWORD 00000000h
 2Ah  DWORD pointer to SYSTEM_LOG_EXIT routine, FFFFFFFFh= don't log errors
 2Eh  DWORD 00000000h
 32h  BYTE  maximum TPs
 33h  BYTE  00h stop QUEUE_ALLOCATEs
    01h resume QUEUE_ALLOCATEs
 34h  DWORD pointer to LU_LU_PASSWORD_EXIT routine, FFFFFFFFh = no exit
 38h  DWORD 00000000h
--------N-6804--------------------------------
INT 68 - APPC/PC - TRANSACTION PROCESSING
  AH = 04h
  DS:DX -> control block (see #03702)
Return: control block updated
```

```
Format of APPC/PC control block:
Offset  Size  Description (Table 03702)
 00h 12 BYTEs reserved
 0Ch  WORD  verb (action)
    2500h TP_ENDED
    2900h TP_VALID
 0Eh  6 BYTEs 00h
 14h  DWORD (big-endian) return code (see #03668)
 18h  WORD  00h
 1Ah  8 BYTEs (big-endian) TP_ID
 22h  DWORD -> CREATE_TP record (only if verb = 2900h)
--------N-6805-------------------------------
INT 68 - APPC/PC - TRANSFER MESSAGE DATA
  AH = 05h
  DS:DX -> control block (see #03703)
Return: control block updated

Format of APPC/PC "Transfer Message Data" control block:
Offset  Size  Description (Table 03703)
 00h 12 BYTEs reserved
 0Ch  WORD  1C00h (verb "Transfer Message Data")
 0Eh  BYTE  data type
    00h user defined
    01h NMVT
    02h alert subvectors
    03h PDSTATS subvectors
 0Fh  5 BYTEs 00h
 14h  DWORD (big-endian) return code (see #03668)
 18h 12 BYTEs 00h
 24h  BYTE  flags (see #03704)
 25h  BYTE  00h
 26h  WORD  length of data
 28h  N BYTEs data

Bitfields for APPC/PC message transfer flags:
Bit(s)  Description (Table 03704)
 0  don't add correlation subvector
 1  don't add product set ID subvector
 2  don't do SYSLOG
 3  don't send SSCP_PU_SESSION
```

```
SeeAlso: #03703
--------N-6806--------------------------------
INT 68 - APPC/PC - CHANGE NUMBER OF SESSIONS
  AH = 06h
  DS:DX -> control block (see #03705)
Return: control block updated

Format of APPC/PC "Change Number of Sessions" control block:
Offset  Size  Description (Table 03705)
 00h 12 BYTEs reserved
 0Ch  WORD  1500h (verb "Change Number of Sessions")
 0Eh  6 BYTEs 00h
 14h  WORD  (big-endian) primary return code (see #03681)
 16h  DWORD (big-endian) secondary return code (see #03668,#03706)
 1Ah  8 BYTEs (big-endian) logical unit ID
 22h  8 BYTEs blanks
 2Ah  8 BYTEs (big-endian) partner logical unit name
 32h  8 BYTEs (big-endian) mode name
 3Ah  BYTE  bit 7: use MODE_NAME_SELECT_ALL rather than MODE_NAME
    bit 6: set negotiable values
 3Bh  BYTE  partner logical unit mode session limit
 3Ch  BYTE  minimum CONWINNERS_SOURCE
 3Dh  BYTE  maximum CONWINNERS_TARGET
 3Eh  BYTE  automatic activation
 3Fh  BYTE  00h
 40h  BYTE  flags
    bit 7: drain target
    bit 6: drain source
    bit 5: target responsible, not source

(Table 03706)
Values for secondary return code (see also AH=01h/SF=1B00h):
 0000h  accepted
 0001h  negotiated
 0003h  bad logical unit ID
 0004h  allocation failure, no retry
 0005h  allocation failure, retry
 0151h  can't raise limits
 0153h  all modes must reset
 0154h  bad SNASVCMG limits
 0155h  minimum greater than total
```

```
 0156h  mode closed (primary return code = 1)
  CNOS mode closed (primary return code = 18h)
 0157h  bad mode name (primary return code = 1)
  CNOS bad mode name (primary return code = 18h)
 0159h  reset SNA drains
 015Ah  single not SRC response
 015Bh  bad partner logical unit
 015Ch  exceeds maximum allowed
 015Dh  change SRC drains
 015Eh  logical unit detached
 015Fh  CNOS command race reject
--------N-6807-------------------------------
INT 68 - APPC/PC - PASSTHROUGH
  AH = 07h
  DS:DX -> control block (format depends on application subsystem)
Return: control block updated
SeeAlso: AH=FFh
----------684300-----------------------------
INT 68 U - ??? - INSTALLATION CHECK???
  AX = 4300h
Return: AX = F386h if ???
  ???
Note: called by Novell DOS 7.0 EMM386.EXE
SeeAlso: AX=4400h,INT 41/AX=004Fh
----------684400-----------------------------
INT 68 U - ???
  AX = 4400h
  BX = ???
  CX = ???
  DX = ???
  DS:SI = real-mode address of protected-mode GDT
  ES:DI = real-mode address of protected-mode IDT
Return: ???
Note: called by Novell DOS 7.0 EMM386.EXE if AX=4300h returns AX=F386h
SeeAlso: AX=4300h
--------W-6847-------------------------------
INT 68 - MS Windows debugging kernel - OUTPUT STRING
  AH = 47h
  ES:SI -> string
Notes:  output a string (to inform a debugger of some events)
  KERNEL outputs "Windows Kernel Entry\r\n" on startup
```

```
SeeAlso: INT 41/AX=0012h
--------N-68FA-------------------------------
INT 68 - APPC/PC - ENABLE/DISABLE APPC
  AH = FAh
  AL bit 0 = new state (0 enable, 1 disable)
SeeAlso: AH=FDh,INT 68"Novell"
--------N-68FB-------------------------------
INT 68 - APPC/PC - CONVERT
  AH = FBh
  DS:DX -> control block (see #03707)
Return: control block updated

Format of APPC/PC "CONVERT" control block:
Offset  Size  Description (Table 03707)
 00h 12 BYTEs reserved
 0Ch  WORD  1A00h (verb "CONVERT")
 0Eh  6 BYTEs 00h
 14h  DWORD (big-endian) return code
 18h  BYTE  conversion
    00h ASCII to EBCDIC
    01h EBCDIC to ASCII
 19h  BYTE  character set
    00h AE
    01h A
    02h G
 1Ah  WORD  length of string to convert
 1Ch  DWORD pointer to source
 20h  DWORD pointer to target
--------N-68FC-------------------------------
INT 68 - APPC/PC - ENABLE/DISABLE MESSAGE TRACING
  AH = FCh
  AL = new state
      00h disable tracing
      01h enable tracing
    DX = number of bytes to keep (0=all)
SeeAlso: AH=FDh,AH=FEh
--------N-68FD-------------------------------
INT 68 - APPC/PC - ENABLE/DISABLE API VERB TRACING
  AH = FDh
  AL = new tracing state (00h disabled, 01h enabled)
SeeAlso: AH=FAh,AH=FCh,AH=FEh
```

```
--------N-68FE-------------------------------
INT 68 - APPC/PC - SET TRACE DESTINATION
  AH = FEh
  AL = trace destinations (see #03708)
  DS:DX -> trace stats record if AL bit 0 set (see #03709)
SeeAlso: AH=FCh,AH=FDh

Bitfields for trace destinations:
Bit(s)  Description (Table 03708)
 0  storage (DS:DX -> trace stats record)
 1  display
 2  file (trace written to file OUTPUT.PC)
 3  printer

Format of APPC/PC Trace Statistics Record:
Offset  Size  Description (Table 03709)
 00h  DWORD pointer to storage trace buffer
 04h  WORD  max number of 80-byte records in trace
 06h  WORD  (high-order byte first!) current record number (must init to 0)
 08h  DWORD (high-order byte first!) number of records written (init to 0)
 0Ch  DWORD reserved
Note: do not move record while trace is active
--------N-68FF-------------------------------
INT 68 - APPC/PC - SET PASSTHROUGH
  AH = FFh
  DS:DX -> passthrough exit routine
SeeAlso: AH=07h,INT 68"Novell"
--------b-69---------------------------------
INT 69 - Zenith AT BIOS - ???
Note: called by INT 09 handler
--------h-69---------------------------------
INT 69 C - HP Vectra AT - IRQ17 - KEYBOARD OUTPUT-BUFFER-FULL SERVICE ROUTINE
SeeAlso: INT 09"IRQ1",INT 68"HP Vectra",INT 6A"HP Vectra"
--------N-690100-----------------------------
INT 69 - DECnet DOS CTERM - INSTALLATION CHECK
  AX = 0100h
Return: AL = FFh if present
SeeAlso: AX=010Fh
--------N-690101-----------------------------
INT 69 - DECnet DOS CTERM - SEND BYTE
  AX = 0101h
```

```
  BL = character
  DX = session handle
Return: AH >= 80h on error
SeeAlso: AX=0102h
--------N-690102-----------------------------
INT 69 - DECnet DOS CTERM - READ BYTE
  AX = 0102h
  DX = session handle
Return: AH >= 80h on error
  AH < 80h if successful
      AL = character
SeeAlso: AX=0101h
--------N-690103-----------------------------
INT 69 - DECnet DOS CTERM - STATUS
  AX = 0103h
  DX = session handle
Return: AH status flags (see #03710)
  AL = reason code if DECnet error (see #03711)
SeeAlso: AX=0104h

Bitfields for DECnet DOS CTERM status flags:
Bit(s)  Description (Table 03710)
 7  session has been aborted
 6  DECnet error
 1  trace data available
 0  receive data available

(Table 03711)
Values for reason code:
 00h  normal disconnect
 01h  unknown message from host
 02h  protocol violation from host
 03h  could not process the initiate message
 04h  error receiving message from host
 05h  error sending message to host
 06h  error checking for message from host
 07h  remote system does not support CTERM
 08h  remote system does not support correct protocol version
 09h  did not receive BIND message from host
 0Ah  could not send BIND message to host
 0Bh  no more sessions available
```

```
 0Ch  session does not exist
 0Dh  not enough memory to complete operation
 0Eh  connection has broken
Index:  error codes;DECnet DOS CTERM|DECnet DOS CTERM;error codes
--------N-690104-----------------------------
INT 69 - DECnet DOS CTERM - DECnet STATUS
  AX = 0104h
  DX = session handle
Return: AX = reason code (see #03711)
Note: use this call when AX=0103h returns a DECnet error
SeeAlso: AX=0103h
--------N-690105-----------------------------
INT 69 - DECnet DOS CTERM - OPEN SESSION
  AX = 0105h
  DS:BX -> ASCIZ node name
  ES:DX -> buffer for session control block (see #03727 at INT 6A/AH=D0h)
Return: AX <= 0 on error
  AX > 0 session handle
SeeAlso: AX=0103h,AX=0106h,AX=010Ah
--------N-690106-----------------------------
INT 69 - DECnet DOS CTERM - CLOSE SESSION
  AX = 0106h
  DX = session handle
Return: AH = status
      00h good close
      other error code (see #03711)
SeeAlso: AX=0103h,AX=0105h
--------N-69010A-----------------------------
INT 69 - DECnet DOS CTERM - GET SESSION CONTROL BLOCK SIZE
  AX = 010Ah
Return: AX = length of session control block in bytes
SeeAlso: AX=0105h
--------N-69010B-----------------------------
INT 69 - DECnet DOS CTERM - GET DECnet SOCKET
  AX = 010Bh
  DX = session handle
Return: AX > 0  DECnet socket for the session
  AX = 0  no match for handle
--------N-69010F-----------------------------
INT 69 - DECnet DOS CTERM - DEINSTALL CTERM
  AX = 010Fh
```

```
Return: AH = status
      00h successful uninstall
      other error code (see #03711)
Note: CTERM must have been the last TSR loaded in order to deinstall it
SeeAlso: AX=0100h
Index:  uninstall;DECnet DOS CTERM
--------N-690A-------------------------------
INT 69 - DECnet DOS 2.1+ - DATA LINK LAYER
  AH = 0Ah
  AL = function number (see #03712)
  ES:BX -> Datalink Communication Block (see #03714)
Return: AX = status (see #03713)
SeeAlso: INT 6D"DECnet"

(Table 03712)
Values for DECnet DOS Data Link Layer function:
 00h  initialize
 01h  open portal
 02h  close portal
 03h  enable multicast address
 04h  disable multicast address
 05h  transmit
 06h  request transmit buffer
 07h  deallocate transmit buffer
 08h  read channel status
 09h  read datalink portal list
 0Ah  read information about a datalink portal
 0Bh  read and/or clear counters
 0Ch  request to boot from a network server
 0Dh  enable Ethernet channel
 0Eh  disable Ethernet channel
 0Fh  start MOP/send a System ID message
 10h  stop MOP
 11h  get DECPARM
 12h  set DECPARM
 13h  external loopback

(Table 03713)
Values for DECnet DOS Data Link Layer status:
 00h  successful
 01h  hardware failed to initialize
```

```
 02h  channel state was not off (must be off to execute that command)
 03h  channel state is off (must be on to execute that command)
 04h  address not set
 05h  hardware missing
 06h  buffer too small
 07h  no more buffers available
 08h  no more resources available
 09h  promiscuous receiver active
 0Ah  non exclusive
 0Bh  unrecognized portal
 0Ch  protocol type in use
 0Dh  not a valid Multicast address
 0Eh  outstanding calls
 0Fh  hardware doesn't support receiving bad frames
 10h  none outstanding
 11h  no events
 12h  broken
 13h  buffer quota exceeded
 14h  already initialized
 15h  loopback failure
Index:  error codes;DECnet DOS|DECnet DOS;error codes

Format of Datalink Communication Block:
Offset  Size  Description (Table 03714)
 00h  WORD  portal ID
 02h  6 BYTEs source address
 08h  6 BYTEs destination address
 0Eh  DWORD buffer pointer
 12h  WORD  buffer length
 14h  WORD  operation
 16h  BYTE  pad flag (used on open)
    00h no pad
    01h pad
 17h  BYTE  mode flag (used on open)
    00h 802.3
    01h Ethernet
    02h promiscuous
 18h  DWORD line status change function
 1Ch  DWORD received data function
 20h  DWORD transmitted data function
 24h  BYTE  maximum outstanding transmits/receives
```

```
 25h  2 BYTEs protocol type
 27h  WORD  buffers lost
--------N-694001-----------------------------
INT 69 - 10NET v5.0 - SYSSVC.COM - ???
  AX = 4001h
Return: CF clear
  AX = 0000h
  ES:SI -> ???
InstallCheck: test for the signature "SYSV" immediately before the interrupt
    handler
Range:  INT 60 to INT 7F, selected by configuration
SeeAlso: AX=4002h
Index:  installation check;10NET SYSSVC
--------N-694002-----------------------------
INT 69 - 10NET v5.0 - SYSSVC.COM - ???
  AX = 4002h
  ???
Return: ???
InstallCheck: test for the signature "SYSV" immediately before the interrupt
    handler
Range:  INT 60 to INT 7F, selected by configuration
--------N-694101-----------------------------
INT 69 - 10NET v5.0 - SYSSVC.COM - ???
  AX = 4101h
Return: CF clear
  ES:SI -> ???
Range:  INT 60 to INT 7F, selected by configuration
SeeAlso: AX=4102h,AX=4103h,AX=4104h
--------N-694102-----------------------------
INT 69 - 10NET v5.0 - SYSSVC.COM - ???
  AX = 4102h
  ???
Return: ???
--------N-694103-----------------------------
INT 69 - 10NET v5.0 - SYSSVC.COM - ???
  AX = 4103h
  ???
Return: ???
--------N-694104-----------------------------
INT 69 - 10NET v5.0 - SYSSVC.COM - ???
  AX = 4104h
```

```
  ???
Return: ???
--------N-6942-------------------------------
INT 69 - 10NET v5.0 - SYSSVC.COM - ???
  AH = 42h
  AL = function (01h-14h)
  ???
Return: ???
Range:  INT 60 to INT 7F, selected by configuration
--------N-6943-------------------------------
INT 69 - 10NET v5.0 - SYSSVC.COM - ???
  AH = 43h
  AL = function (01h-05h)
  ???
Return: ???
--------N-6944-------------------------------
INT 69 - 10NET v5.0 - SYSSVC.COM - ???
  AH = 44h
  AL = function (01h-03h)
  ???
Return: ???
Range:  INT 60 to INT 7F, selected by configuration
--------N-6949-------------------------------
INT 69 - 10NET v5.0 - SYSSVC.COM - BUG
  AH = 49h
Note: due to a fencepost error, this function branches to hyperspace
SeeAlso: AX=4001h,AH=FFh
--------G-696996-----------------------------
INT 69 - ISR.COM v1.00 - SPECIFY INTERRUPT HANDLER
  AX = 6996h
  DS:DX -> interrupt handler or 0000h:0000h to disable
Return: AX = 9669h
Program: ISR (Interrupt Service Reflector) is a TSR by Rich Bono which permits
    a program to provide hardware interrupt handlers even while being
    debugged with a debugger that swaps interrupt vectors during
    debugging.
Note: the interrupt vector which is to be reflected is set at installation
    time and cannot be changed
--------N-69FF-------------------------------
INT 69 - 10NET v5.0 - SYSSVC.COM - SIGNAL SYSTEM ERROR
  AH = FFh
```

```
Return: never???
Desc: displays "System Error" message and register dump, then halts system
InstallCheck: test for the signature "SYSV" immediately before the interrupt
    handler
Range:  INT 60 to INT 7F, selected by configuration
SeeAlso: AX=4001h,AH=49h
--------U-6A---------------------------------
INT 6A - OPTHELP.COM
Program: OPTHELP is an optionally-resident help system for SLR Systems's OPTASM
    assembler
Range:  INT 60h to INT 7Fh, selected by configuration
--------N-6A---------------------------------
INT 6A - DECnet DOS - LOCAL AREA TRANSPORT PROGRAM - INSTALLATION CHECK
InstallCheck: test for a signature area immediately preceding the interrupt
    handler (see #03715)
SeeAlso: AH=01h/DH=FFh,INT 6B"DECnet",INT 6D"DECnet"
Index:  installation check;DECnet DOS Local Area Transport

Format of DECnet DOS signature area:
Offset  Size  Description (Table 03715)
 -5 BYTE  major version number
 -4 BYTE  minor version number
 -3   3 BYTEs signature (ASCII "LAT")
--------h-6A---------------------------------
INT 6A C - HP Vectra AT - IRQ18 - RESERVED HARDWARE INTERRUPT
SeeAlso: INT 0A"IRQ2",INT 69"HP Vectra",INT 6B"HP Vectra"
--------N-6A0000-----------------------------
INT 6A U - Super-TCP DOS TSR Kernel v3.57 - INSTALLATION CHECK
  AX = 0000h
Return: AX = 4357h ('CW')
Program: Super-TCP is a TCP/IP protocol stack by Frontier Technologies Corp.
Note: an alternate installation check is to test for the ASCIZ signature
    "FTC Super-TCP" three bytes past the interrupt handler
SeeAlso: AX=0001h,AX=0002h,AX=000Fh,AX=0010h,INT 21/AH=3Fh"BW-TCP"
SeeAlso: INT 61"FTP Software",INT 62/AH=00h"ETHDEV"
--------N-6A0001-----------------------------
INT 6A U - Super-TCP DOS TSR Kernel v3.57 - ???
  AX = 0001h
  BH = function number
      01h ???
    DS:SI -> ??? 24-byte record1 (see #03717)
```

```
    ES:DI -> buffer containing ???
      02h ???
    DS:SI -> ??? 18-byte record2 (see #03718)
    ES:DI -> buffer containing ???
      04h ???
    BL = subfunction
        01h
      DS:SI -> ??? 28-byte record3 (see #03719)
      ES:DI -> buffer containing ???
        02h
      DS:SI -> ??? 28-byte record3 (see #03719)
      ES:DI -> buffer containing ???
        03h
      DS:SI -> ??? 28-byte record3 (see #03719)
        else Return: AX = 0005h
      05h ???
    DS:SI -> ??? 20-byte record4 (see #03720)
    ES:DI -> buffer containing ???
      06h ???
    BL = subfunction
        01h
      DS:SI -> ??? 40-byte record5 (see #03721)
        02h
      DS:SI -> ??? 20-byte record6 (see #03722)
      ES:DI -> ???
        03h
      DS:SI -> ??? 20-byte record6 (see #03722)
        04h
      DS:SI -> ??? 46-byte record7 (see #03723)
        else Return: AX = 0005h
      11h ???
    DS:SI -> ??? 28-byte record8 (see #03724)
    ES:DI -> ???
Return: AX = function status (see #03716)
SeeAlso: AX=0000h

(Table 03716)
Values for Super-TCP function status:
 0000h  successful
 0005h  unsupported function
 000Ah  out of memory
```

```
Format of record1:
Offset  Size  Description (Table 03717)
 00h  4 BYTEs ???
 04h  WORD  size of ES:DI buffer
 06h 18 BYTEs ???

Format of record2:
Offset  Size  Description (Table 03718)
 00h  4 BYTEs ???
 04h  WORD  size of ES:DI buffer
 06h 12 BYTEs ???

Format of record3:
Offset  Size  Description (Table 03719)
 00h  2 BYTEs ???
 02h  WORD  ???
 04h  WORD  size of ES:DI buffer
 06h  WORD  ???
 08h  WORD  operation number (for function 0401h)
 0Ah  DWORD -> ???
 0Eh  WORD  (ret) ???
 10h 12 BYTEs ???

Format of record4:
Offset  Size  Description (Table 03720)
 00h  4 BYTEs ???
 04h  WORD  size of ES:DI buffer
 06h 14 BYTEs ???

Format of record5:
Offset  Size  Description (Table 03721)
 00h  BYTE  operation??? (00h-07h)
 01h  BYTE  ???
 02h  WORD  (ret) ???
 04h  DWORD -> ???
 08h  4 BYTEs ???
 0Ch  DWORD -> ??? or 0000h:0000h
 10h 16 BYTEs ???
 20h  DWORD ???
 24h  4 BYTEs ???
```

```
Format of record6:
Offset  Size  Description (Table 03722)
 00h  4 BYTEs ???
 04h  WORD  size of ES:DI buffer
 06h 14 BYTEs ???

Format of record7:
Offset  Size  Description (Table 03723)
 00h  WORD  ???
 02h  WORD  ???
 04h  WORD  ???
 06h 40 BYTEs ???

Format of record8:
Offset  Size  Description (Table 03724)
 00h  4 BYTEs ???
 04h  WORD  size of ES:DI buffer
 06h  6 BYTEs ???
 0Ch  WORD  (ret) ???
 0Eh  WORD  operation??? (01h-03h)
 10h 12 BYTEs ???
--------N-6A0002-----------------------------
INT 6A U - Super-TCP DOS TSR Kernel v3.57 - ???
  AX = 0002h
  BX = ??? (zero/nonzero)
  CX = ??? identifier (see AX=0004h)
  DS:SI -> 40-byte buffer for ??? or 0000h:0000h
  ES:DI -> buffer for ??? or 0000h:0000h
Return: AX = 0000h (successful) ???
  BL = ???
  BH = ???
  CX = ???
  DX = ???
--------N-6A0003-----------------------------
INT 6A U - Super-TCP DOS TSR Kernel v3.57 - GET ??? DATA AREA
  AX = 0003h
Return: CX:DX -> data area (see #03725)

Format of Super-TCP data area:
Offset  Size  Description (Table 03725)
```

```
 00h  2 BYTEs ???
 02h  DWORD original INT 6A vector
 06h  2 BYTEs ???
 08h 96 BYTEs array of 16 6-byte ???
 68h  WORD  number of elements of above array in use
 6Ah  WORD  ???
  ???
--------N-6A0004-----------------------------
INT 6A U - Super-TCP DOS TSR Kernel v3.57 - ALLOCATE ???
  AX = 0004h
  CX = size in ???
Return: AX = 0000h (successful)
  CX = DX = ???
SeeAlso: AX=0005h,AX=000Fh
--------N-6A0005-----------------------------
INT 6A U - Super-TCP DOS TSR Kernel v3.57 - FREE/CLOSE ???
  AX = 0005h
  CX = ??? identifier (from AX=0004h)
Return: AX = status (0000h successful, FFFFh failed)
SeeAlso: AX=0004h,AX=000Fh
--------N-6A000F-----------------------------
INT 6A U - Super-TCP DOS TSR Kernel v3.57 - FREE/CLOSE ALL ???
  AX = 000Fh
Return: AX = 0000h (successful)
SeeAlso: AX=0000h,AX=0004h,AX=0005h
--------N-6A0010-----------------------------
INT 6A U - Super-TCP DOS TSR Kernel v3.57 - UNINSTALL
  AX = 0010h
Return: AX = status
      0000h successful
      0002h can't uninstall, interrupt vector hooked by another program
Program: Super-TCP is a TCP/IP protocol stack by Frontier Technologies Corp.
Note: if AX is not one of the values listed here on entry, Super-TCP
    returns AX=FFFEh
SeeAlso: AX=0000h
--------N-6A01--DHFF------------------------
INT 6A - DECnet DOS LOCAL AREA TRANSPORT - SEND BYTE
  AH = 01h
  DH = FFh
  AL = character
  DL = handle
```

```
Return: AH >= 80h on error
SeeAlso: AH=02h
--------N-6A02--DHFF-------------------------
INT 6A - DECnet DOS LOCAL AREA TRANSPORT - READ BYTE
  AH = 02h
  DH = FFh
  DL = handle
Return: AH < 80h if successful
      AL = character
  AH >= 80h on error
SeeAlso: AH=01h
--------N-6A03--DHFF-------------------------
INT 6A - DECnet DOS LOCAL AREA TRANSPORT - STATUS
  AH = 03h
  DH = FFh
  DL = handle
Return: AH = status flags (see #03726)

Bitfields for DECnet DOS LAT status flags:
Bit(s)  Description (Table 03726)
 5  transmit buffer empty
 3  session in start state
 2  session not active
 1  unable to queue transmit data
 0  receive data available
--------N-6AD0--DHFF-------------------------
INT 6A - DECnet DOS LOCAL AREA TRANSPORT - OPEN SESSION
  AH = D0h
  DH = FFh
  AL = password flag
       FFh no password
       0Fh password at ES:DI
  ES:BX -> LAT session control block (see #03727)
  ES:DI -> 16-byte blank-padded password (optional)
Return: AH = 00h success
      DL = handle
SeeAlso: AX=D000h

Format of LAT Session Control Block:
Offset  Size  Description (Table 03727)
 00h 18 BYTEs service name
```

```
 12h 18 BYTEs node name (future use)
 24h 18 BYTEs port name (future use)
 36h  DWORD -> session stopped post routine
 3Ah  DWORD -> service table overflow post routine
 3Eh  DWORD -> transmit post routine
 42h  DWORD -> receive post routine
 46h  WORD  session status
    04h circuit failure
    08h stop slot received
---LAT v???---
 48h  WORD  slot state (LAT driver use)
 4Ah  WORD  local credits (LAT driver use)
 4Ch  DWORD -> VCB (LAT driver use)
 50h  WORD  backward slot (LAT driver use)
 52h  WORD  forward slot (LAT driver use)
 54h  WORD  remote slot ID (LAT driver use)
 56h  WORD  local slot ID (LAT driver use)
 58h  WORD  slot byte count (LAT driver use)
 5Ah  BYTE  remote credits (LAT driver use)
 5Bh 255 BYTEs  transmitted data slot
15Ah  BYTE  number of receive data slots (4 recommended)
15Bh  BYTE  number of occupied slots
15Ch  BYTE  index of next receive slot to use
15Dh  BYTE  index of current receive slot
15Eh  WORD  pointer to first received character
160h  N WORDs pointers to receive slots (buffers); each is 259 bytes
     259N BYTEs buffers
Note: set post routines to 0000h:0000h if polled operation will be used
---LAT v4.1.17---
 48h  WORD  session state (LAT driver use)
 4Ah  BYTE  local credits (LAT driver use)
 4Bh  DWORD -> VCB (LAT driver use)
 4Fh  WORD  backward slot (LAT driver use)
 51h  WORD  forward slot (LAT driver use)
 53h  BYTE  remote slot ID (LAT driver use)
 54h  BYTE  local slot ID (LAT driver use)
 55h  BYTE  slot byte count (LAT driver use)
 56h  BYTE  remote credits (LAT driver use)
 57h 255 BYTEs  transmitted data slot
156h  BYTE  number of receive data slots (4 recommended)
157h  BYTE  number of occupied slots
```

```
158h  BYTE  index of next receive slot to use
159h  BYTE  index of current receive slot
15Ah  WORD  pointer to first received character
15Ch  N WORDs pointers to receive slots (buffers); each is 259 bytes
     259N BYTEs buffers
Note: set post routines to 0000h:0000h if polled operation will be used
--------N-6AD000DHFF-------------------------
INT 6A - DECnet DOS LOCAL AREA TRANSPORT - CLOSE SESSION
  AX = D000h
  DH = FFh
  DL = handle
Return: AX = status (see #03728)
SeeAlso: AH=D0h

(Table 03728)
Values for DECnet DOS LAT function status:
 0000h  successful
 0001h  no such session
 0002h  session not running, try again later
--------N-6AD100DHFF-------------------------
INT 6A - DECnet DOS LOCAL AREA TRANSPORT - SEND BREAK
  AX = D100h
  DH = FFh
  DL = handle
Return: AX = 0000h if successful
  AH bit 7 set if unable to send break
--------N-6AD300DHFF-------------------------
INT 6A - DECnet DOS LOCAL AREA TRANSPORT - RESET LAT COUNTERS
  AX = D300h
  DH = FFh
SeeAlso: AX=D400h
--------N-6AD400DHFF-------------------------
INT 6A - DECnet DOS LOCAL AREA TRANSPORT - COPY LAT COUNTERS
  AX = D400h
  DH = FFh
  CX = buffer size
  ES:BX -> buffer for LAT counters
Return: AX = status
      0000h counters copied into buffer
      FFFFh buffer too small
SeeAlso: AX=D300h
```

```
--------N-6AD500DHFF-------------------------
INT 6A - DECnet DOS LOCAL AREA TRANSPORT - GET NEXT LAT SERVICE NAME
  AX = D500h
  DH = FFh
  ES:BX -> 17-byte buffer for name
Return: AH = 00h if successful
     ES:BX buffer filled
  AX = FFFFh if end of table or no name available
Notes:  use this function to get the names of the hosts on the network
  successive calls are necessary to get all names
SeeAlso: AX=D600h/DH=FFh
--------N-6AD600DHFF-------------------------
INT 6A - DECnet DOS LOCAL AREA TRANSPORT - LAT SERVICE TABLE RESET
  AX = D600h
  DH = FFh
Return: AX = number of service table entries
  BX = status
       0000h service table has not overflowed
       FFFFh service table has overflowed
SeeAlso: AX=D500h
--------!---Section-------------------------
```