```
Interrupt List, part 11 of 18
Copyright (c) 1989-1999,2000 Ralf Brown
--------D-26---------------------------------
INT 26 - DOS 1+ - ABSOLUTE DISK WRITE (except partitions > 32M)
  AL = drive number (00h = A:, 01h = B:, etc)
  CX = number of sectors to write (not FFFFh)
  DX = starting logical sector number (0000h - highest sector on drive)
  DS:BX -> data to write
Return: CF clear if successful
  CF set on error
      AH = status (see #02547)
      AL = error code (same as passed to INT 24 in DI)
      AX = 0207h if more than 64K sectors on drive -- use new-style call
  may destroy all other registers except segment registers
Notes:  original flags are left on stack, and must be popped by caller
  this call bypasses the DOS filesystem, though DOS 5+ invalidates any
    disk buffers referencing sectors which are written with this call
  examination of CPWIN386.CPL indicates that if this call fails with
    error 0408h on an old-style (<32M) call, one should retry the
    call with the high bit of the drive number in AL set
  Novell DOS 7 decides whether the old-style or new-style (>32M) version
    of INT 26 must be used solely on the basis of the partition's size,
    thus forcing use of the new-style call even for data in the first
    32M of the partition
  Windows98 will display an error message and deliberately hang the
    system on attempted write to any hard disk if neither bit 7 of the
    Extended Drive Info byte nor bit 6 of "DOS_FLAG" (List-of-Lists+60h)
    is set
  Although all registers except segment registers may be destroyed
   some software depends on some of the registers being preserved.
   For example some Flash disk drivers requires that DX is not trashed.
   DR-DOS 7.03 takes care of this.
BUGS: DOS 3.1 through 3.3 set the word at ES:[BP+1Eh] to FFFFh if AL is an
    invalid drive number
  DR DOS 3.41 will return with a jump instead of RETF, leaving the
    wrong number of bytes on the stack; use the huge-partition version
    (INT 26/CX=FFFFh) for all partition sizes under DR DOS 3.41
  DR DOS 6.0 original releases 05/1991 & 08/1991 reported wrong error
    codes for "drive not ready" and "write protect". This was fixed
    with the DR DOS BDOS patch "PAT321" (1992/02/19, XDIR /C: 947Bh)
    and later "full" rebuilds (see INT21/AX=4452h for details).
```

```
SeeAlso: INT 13/AH=03h,INT 25,INT 26/CX=FFFFh,INT 21/AX=7305h,INT 21/AH=91h"PTS"
--------D-26----CXFFFF----------------------
INT 26 - DOS 3.31+ - ABSOLUTE DISK WRITE (32M-2047M hard-disk partition)
  CX = FFFFh
  AL = drive number (0=A, 1=B, etc)
  DS:BX -> disk write packet (see #02552)
Return: CF clear if successful
  CF set on error
      AH = status (see #02547)
      AL = error code (same as passed to INT 24 in DI)
  may destroy all other registers except segment registers
Notes:  partition is potentially >32M (and requires this form of the call) if
    bit 1 of the device attribute word in the device driver is set
  original flags are left on stack, and must be removed by caller
  this call bypasses the DOS filesystem, though DOS 5+ invalidates any
    disk buffers referencing sectors which are written with this call
  for FAT32 drives (which may be up to 2TB in size), use INT 21/AX=7305h
  Windows98 will display an error message and deliberately hang the
    system on attempted write to any hard disk if neither bit 7 of the
    Extended Drive Info byte nor bit 6 of "DOS_FLAG" (List-of-Lists+60h)
    is set
SeeAlso: INT 13/AH=03h,INT 25/CX=FFFFh,INT 26,INT 21/AX=7305h

Format of disk write packet:
Offset  Size  Description (Table 02552)
 00h  DWORD sector number
 04h  WORD  number of sectors to read
 06h  DWORD transfer address
SeeAlso: #02548
--------G-26---------------------------------
INT 26 - COMTROL HOSTESS i/ISA DEBUGGER - ENTER/EXIT EXTENDED ADDRESSING MODE
  ???
Return: ???
SeeAlso: INT 23"COMTROL",INT 27"COMTROL"
--------D-27---------------------------------
INT 27 - DOS 1+ - TERMINATE AND STAY RESIDENT
  DX = number of bytes to keep resident (max FFF0h)
  CS = segment of PSP
Return: never
Notes:  this is an obsolete call
  INT 22, INT 23, and INT 24 are restored from the PSP
```

```
  does not close any open files
  the minimum number of bytes which will remain resident is 110h for
    DOS 2.x and 60h for DOS 3.0+; there is no minimum for DOS 1.x, which
    implements this service in COMMAND.COM rather than the DOS kernel
SeeAlso: INT 21/AH=31h
--------G-27--------------------------------
INT 27 - COMTROL HOSTESS i/ISA DEBUGGER - INVOKE REMOTE TURBO DEBUGGER KERNEL
  ???
Return: ???
Desc: invoke a copy of the remote Turbo Debugger kernel on the Hostess i
    controller
SeeAlso: INT 20"COMTROL",INT 26"COMTROL"
--------D-28--------------------------------
INT 28 C - DOS 2+ - DOS IDLE INTERRUPT
  SS:SP = top of MS-DOS stack for I/O functions
Return: all registers preserved
Desc: This interrupt is invoked each time one of the DOS character input
    functions loops while waiting for input.  Since a DOS call is in
    progress even though DOS is actually idle during such input waits,
    hooking this function is necessary to allow a TSR to perform DOS
    calls while the foreground program is waiting for user input.  The
    INT 28h handler may invoke any INT 21h function except functions
    00h through 0Ch.
Notes:  under DOS 2.x, the critical error flag (the byte immediately after the
    InDOS flag) must be set in order to call DOS functions 50h/51h from
    the INT 28h handler without destroying the DOS stacks.
  calls to INT 21/AH=3Fh,40h from within an INT 28 handler may not use a
    handle which refers to CON
  at the time of the call, the InDOS flag (see INT 21/AH=34h) is normally
    set to 01h; if larger, DOS is truly busy and should not be reentered
  the default handler is an IRET instruction
  supported in OS/2 compatibility box
  the _MS-DOS_Programmer's_Reference_ for DOS 5.0 incorrectly documents
    this interrupt as superseded
  the performance of NetWare Lite servers (and probably other peer-to-
    peer networks) can be dramatically improved by calling INT 28
    frequently from an application's idle loop
SeeAlso: INT 21/AH=34h,INT 2A/AH=84h,INT 2F/AX=1680h
--------U-289999------------------------------
INT 28 u - PCXDUMP v9.00+ - INSTALLATION CHECK
  AX = 9999h
```

```
Return: AX = AAAAh if installed
      CX = version number * 100 (example: 03A2h = 930 = v9.30)
      DL = interrupt used by the dump function (see #02553)
    (00h if call not available)
      BX = CS of PCXDUMP's INT 28 handler (undocumented)
      ES = segment of PCXDUMP's memory block (v9.30, undocumented)
Program: PCXDUMP is a shareware screen grabber saving in PCX format
Notes:  if DL<>00h a dump can be requested by calling INT DL as shown
    below (the user can choose the interrupt number at installation
    time); if DL=00h the dump function can be called only by hotkeys
    (this is the default)

(Table 02553)
Call PCXDUMP screen-dump function with:
  INT xx
  AX = 1234h
  BX = dump type
      0000h Color dump
      0001h Immediate color dump
      0002h Black/White dump
      0003h Immediate B/W dump
      0004h Inverted B/W dump
      0005h Gray scaled dump
      0006h Inverted gray scaled dump
      0007h Text screen dump to text file
      0008h Text screen dump to ansi file
Return: nothing
Notes:  if BX=0001h, 0003h, 0007h or 0008h the whole screen will be
    dumped; the other valid values will draw a selection frame
    on the screen except in text modes (text modes allow only
    full screen dumps)
  this function doesn't perform the dump, it only requests it;
    the dump will be performed after a few milliseconds if it's
    safe to do so, thus the author recommends putting a 60 ms delay
    after this call
--------D-29---------------------------------
INT 29 C - DOS 2+ - FAST CONSOLE OUTPUT
  AL = character to display
Return: nothing
  BX may be destroyed by some versions of DOS 3.3
Notes:  automatically called when writing to a device with bit 4 of its device
```

```
    driver header set (see also INT 21/AH=52h)
  COMMAND.COM v3.2 and v3.3 compare the INT 29 vector against the INT 20
    vector and assume that ANSI.SYS is installed if the segment is larger
  the default handler under DOS 2.x and 3.x simply calls INT 10/AH=0Eh
  the default handler under DESQview 2.2 understands the <Esc>[2J
    screen-clearing sequence, calls INT 10/AH=0Eh for all others
SeeAlso: INT 21/AH=52h,INT 2F/AX=0802h,INT 79"AVATAR.SYS"
--------U-29E60DCL0E------------------------
INT 29 - ShowGFX - INSTALLATION CHECK
  AX = E60Dh
  CL = 0Eh
  DX = C0DEh
Return: DX = DEC0h
Program: ShowGFX is a PCBoard graphics driver by Solar Designer
--------N-2A00-------------------------------
INT 2A - NETWORK - INSTALLATION CHECK
  AH = 00h
Return: AH <> 00h if installed
  CF set if NetWare v2.15 NetBIOS emulator installed
Note: supported by PC LAN Program, LAN Manager, LANtastic, NetWare, 10NET,
    etc.
SeeAlso: INT 5C"NetBIOS"
--------N-2A0000-----------------------------
INT 2A - AT&T Starlan Extended NetBIOS (var length names) - INSTALLATION CHECK
  AX = 0000h
Return: AH = DDh
SeeAlso: INT 5B"Extended NetBIOS"
--------N-2A01-------------------------------
INT 2A - NETWORK (Microsoft,LANtastic) - EXECUTE NETBIOS REQUEST,NO ERROR RETRY
  AH = 01h
  ES:BX -> NCB (see #03249 at INT 5C"NetBIOS")
Return: AL = NetBIOS error code
  AH = status
       00h no error
       01h error occurred
SeeAlso: AH=04h,AX=0500h,INT 5B"Extended NetBIOS",INT 5C"NetBIOS"
--------N-2A02-------------------------------
INT 2A - NETWORK (Microsoft) - SET NET PRINTER MODE
  AH = 02h
  ???
Return: ???
```

```
--------N-2A0300-----------------------------
INT 2A - NETWORK - CHECK DIRECT I/O
  AX = 0300h
  DS:SI -> ASCIZ device name (may be full path or only drive specifier--
      must include the colon)
Return: CF clear if direct physical addressing (INT 13,INT 25) permissible
  CF set if access via files only
Notes:  do not use direct disk accesses if this function returns CF set or the
    device is redirected (INT 21/AX=5F02h)
  use AH=00h to determine whether the network is installed; if not,
    direct physical access is allowed
  may take some time to execute, so programs which need to check
    frequently should save the result of the first call
  this function is called by the DOS kernel on INT 25 and INT 26
  supported by PC LAN Program, LAN Manage, LANtastic, NetWare, 10NET,
    etc.
SeeAlso: INT 13/AH=02h,INT 13/AH=03h,INT 25,INT 26,INT 21/AX=5F02h
--------N-2A04-------------------------------
INT 2A - NETWORK - EXECUTE NetBIOS REQUEST
  AH = 04h
  AL = error retry
       00h automatically retry request on errors 09h, 12h, and 21h
       (see #03248 at INT 5C"NetBIOS")
       01h no retry
       02h ???
  ES:BX -> Network Control Block (see #03249 at INT 5C"NetBIOS")
Return: AX = 0000h if successful
  AH = 01h on error
       AL = error code
Notes:  invokes either INT 5B or INT 5C as appropriate
  supported by PC LAN Program, LANtastic, LAN Manager, NetWare, 10NET,
    etc.
  NetWare 2.15 NetBIOS emulator returns CF clear if successful, CF set
    on error
  PC LAN Program defines any non-zero return value in AH as an error
    indicator for subfunction 00h, and any non-zero return value in AX
    as an error indicator for subfunction 01h
SeeAlso: AH=00h,AH=01h,AX=0500h,INT 5B"Extended NetBIOS",INT 5C"NetBIOS"
--------N-2A0500-----------------------------
INT 2A - NETWORK - GET NETWORK RESOURCE AVAILABILITY
  AX = 0500h
```

```
Return: AX reserved
  BX = number of network names available
  CX = number of network control blocks available
  DX = number of network sessions available
Notes:  supported by PC LAN Program, LAN Manager, LANtastic, NetWare, 10NET,
    etc.
  the application should call this function before using any network
    resources, and maintain its own count to avoid exceeding the
    network's resource limits
SeeAlso: AH=00h,AH=01h,AH=04h,INT 5C"NetBIOS"
--------N-2A06--------------------------------
INT 2A - NETBIOS, LANtastic - NETWORK PRINT-STREAM CONTROL
  AH = 06h
  AL = function
       01h set concatenation mode
    all printer output put in one job until return to DOS prompt
       02h set truncation mode (default)
    printer open/close or BIOS/DOS output switch starts new job
       03h flush printer output and start new print job
Return: CF set on error
       AX = error code
  CF clear if successful
Notes:  subfunction 03h is equivalent to Ctrl/Alt/keypad-*
  supported by PC LAN Program, LANtastic, NetWare, 10NET, etc.
  LANtastic v4.x no longer supports this call
  this function sets the printer mode for all redirected printers
SeeAlso: INT 21/AX=5D08h,INT 21/AX=5D09h,INT 2F/AX=1125h
--------N-2A07--------------------------------
INT 2A U - PC Network v1.00 - RECEIVER.COM - ???
  AH = 07h
  ???
Return: ???
Program: PC Network is an early networking package which was renamed the
    IBM PC Local Area Network Program (PC LAN Program) as of v1.10
SeeAlso: AH=86h
--------N-2A2001------------------------------
INT 2A - MS Networks or NETBIOS - ???
  AX = 2001h
  ???
Return: ???
Note: intercepted by DESQview 2.x
```

```
--------N-2A2002-----------------------------
INT 2A - NETWORK - ???
  AX = 2002h
  ???
Return: ???
Note: called by MS-DOS 3.30-6.00 APPEND
--------N-2A2003-----------------------------
INT 2A - NETWORK - ???
  AX = 2003h
  ???
Return: ???
Note: called by MS-DOS 3.30-6.00 APPEND
--------N-2A4147DX0000----------------------
INT 2A U - NetSoft DOS-NET v1.20+ - INSTALLATION CHECK
  AX = 4147h ('AG')
  DX = 0000h
Return: DX = 4147h if installed
      DS:SI -> configuration data (see #02554)
Program: DOS-NET is a shareware networking package by Albert Graham
Note: this call is supported by CLIENT.COM, SERVER.COM, ROUTER.COM, and
    NETDOS.COM
SeeAlso: INT 65/DX=4147h,INT 65/DX=4741h

Format of DOS-NET v1.20 configuration data area:
Offset  Size  Description (Table 02554)
 00h  BYTE  ???
 01h  BYTE  interrupt number used by DOS-NET APIs
 02h  WORD  function number to place in AX for above interrupt
 04h  BYTE  minor version as two BCD digits (e.g. 20h for v1.20)
 05h  BYTE  major version number (01h for v1.20)
 06h  2 BYTEs ???
 08h  WORD  ??? (used by ARCNET.COM)
    bit 15: ??? (set by MACTEST.COM)
 0Ah  WORD  ??? (used by NDIS.COM and ODI.COM)
 0Ch 22 BYTEs ???
 24h  DWORD -> ??? function (set by PROTECT.COM)
 28h 12 BYTEs ???
 34h  DWORD -> ??? function (set by FASTVIEW.COM)
 38h  DWORD -> ??? function (set by FASTVIEW.COM)
 58h  DWORD -> ??? (offsets 04h and 1Ah from value are used by NETFILES)
  ???
```

```
 7Ch  WORD  ???
 7Eh  WORD  ??? (may be high half of a DWORD at 7Ch)
  ???
 A8h  DWORD -> ??? (used by SM.COM)
  ???
114h  WORD  ???
116h  WORD  ??? (may be high half of a DWORD at 114h)
  ???
1BDh  BYTE  ??? flags
    bit 0: ???
    bit 6: ???
  ???
1E1h  BYTE  ???
  ???
208h  WORD  ??? (used by SM.COM, MACTEST)
282h  WORD  ???
284h  2 BYTEs ???
286h  WORD  ??? flags
    bit 0: ???
  ???
31Eh  WORD  ???
320h  WORD  ??? (used by NDIS.COM and ODI.COM)
322h  8 BYTEs ???
32Ah  WORD  ??? (used by NDIS.COM and ODI.COM)
  ???
33Eh  4 BYTEs ??? (used by ODI.COM)
342h  N BYTEs ??? (used by NDIS.COM)
  ???
3CFh  BYTE  ??? flags
    bit 2: ???
3D2h  BYTE  installed-component flags
    bit 0: PROTECT installed
    bit 1: NETCACHE installed
    bit 3: SM.COM installed
    bit 7: NETDEBUG installed
3D3h  BYTE  installed-component flags
    bit 0: NETFILES installed
    bit 6: FASTVIEW installed
  ???
3FFh  BYTE  ??? (used by NDIS.COM)
400h  BYTE  ???
```

```
401h  BYTE  ??? (used by SM.COM)
402h  BYTE  ??? (used by SM.COM)
  ???
448h  BYTE  ???
449h  BYTE  ??? (used by MACTEST)
44Ah  BYTE  ??? (used by PROTECT)
44Bh  BYTE  ???
44Ch  BYTE  ???
44Dh  BYTE  ??? (used by SM.COM)
44Eh  BYTE  ??? (used by SM.COM, MACTEST)
  ???
--------N-2A7802-----------------------------
INT 2A - NETWORK - PC LAN PROG v1.31+ - GET LOGGED ON USER NAME
  AX = 7802h
  ES:DI -> 8-byte buffer to be filled
Return: AL = 00h if no user logged on to Extended Services
  AL <> 00h if user logged on to Extended Services
      buffer at ES:DI filled with name, padded to 8 chars with blanks.
--------D-2A80------------------------------
INT 2A CU - NETWORK - BEGIN DOS CRITICAL SECTION
  AH = 80h
  AL = critical section number (00h-0Fh) (see #02555)
Notes:  normally hooked to avoid interrupting a critical section, rather than
    called
  the handler should ensure that none of the critical sections are
    reentered, usually by suspending a task which attempts to reenter
    an active critical section
  the DOS kernel does not invoke critical sections 01h and 02h unless it
    is patched.  DOS 3.1+ contains a zero-terminated list of words
    beginning at offset -11 from the Swappable Data Area
    (see #01687 at INT 21/AX=5D06h); each word contains the offset within
    the DOS data segment of a byte which must be changed from C3h (RET)
    to 50h (PUSH AX) under DOS 3.x or from 00h to a nonzero value under
    DOS 4.0+ to enable use of critical sections.  For DOS 4.0+, all
    words in this list point at the byte at offset 0D0Ch.
  MS Windows patches the DOS kernel's calls to INT 2A/AH=80h-81h into
    far calls to its own handler, and does not reflect the calls back
    to INT 2A unless SYSTEM.INI contains ReflectDOSInt2A=1 or
    ModifyDOSInt2A=0 in the [386Enh] section
  Novell NETX does not issue INT 2A/AH=80h and INT 2A/AH=81h calls when
    it intercepts INT 21 calls and processes them itself
```

```
SeeAlso: AH=81h,AH=82h,AX=8700h,INT 21/AX=5D06h,INT 21/AX=5D0Bh

(Table 02555)
Values for DOS critical section number:
 01h  DOS kernel, SHARE.EXE, DOSMGR
  apparently for maintaining the integrity of DOS/SHARE/NET
    data structures
 02h  DOS kernel, DOSMGR
  ensures that no multitasking occurs while DOS is calling an
    installable device driver
 05h  network redirector
 06h  DOS 4.x only IFSFUNC
 08h  ASSIGN.COM
 0Ah  MSCDEX, CORELCDX
 0Fh  IBM PC LAN server (while intercepting INT 10/AH=06h,07h,0Eh)
--------D-2A81--------------------------------
INT 2A CU - NETWORK - END DOS CRITICAL SECTION
  AH = 81h
  AL = critical section number (00h-0Fh) (see #02555)
Notes:  normally hooked rather than called
  the handler should reawaken any tasks which were suspended due to an
    attempt to enter the specified critical section
  MS Windows patches the DOS kernel's calls to INT 2A/AH=80h-81h into
    far calls to its own handler, and does not reflect the calls back
    to INT 2A unless SYSTEM.INI contains ReflectDOSInt2A=1 or
    ModifyDOSInt2A=0 in the [386Enh] section
SeeAlso: AH=80h,AH=82h,AX=8700h
--------D-2A82--------------------------------
INT 2A CU - NETWORK - END DOS CRITICAL SECTIONS 0 THROUGH 7
  AH = 82h
Notes:  called by the INT 21h function dispatcher for function 0 and functions
    greater than 0Ch except 59h, and on process termination
  the handler should reawaken any tasks which were suspended due to an
    attempt to enter one of the critical sections 0 through 7
SeeAlso: AH=81h
--------N-2A84--------------------------------
INT 2A CU - NETWORK - KEYBOARD BUSY LOOP
  AH = 84h
Notes:  similar to DOS's INT 28h, called from inside the DOS keyboard input
    loop (i.e. INT 21/AH=07h or INT 21/AH=08h) to allow the network
    software to process requests
```

```
  Novell DOS 7+ calls this function with AX=8400h from inside of
    the keyboard input loop.
SeeAlso: INT 28
--------N-2A86-------------------------------
INT 2A U - PC Network v1.00 - RECEIVER.COM - ???
  AH = 86h
  ???
Return: ???
SeeAlso: AH=07h,AH=C4h
--------P-2A8700-----------------------------
INT 2A CU - PRINT - BEGIN BACKGROUND PRINTING
  AX = 8700h
  CF clear
Return: CF clear if OK to print in background now
  CF set if background printing not allowed at this time
Desc: used to inform interested programs that PRINT is about to start its
    background processing, and allow those programs to postpone the
    processing if necessary
Notes:  when PRINT gains control and wants to begin printing, it calls this
    function.  If CF is clear on return, PRINT begins its background
    processing, and calls AX=8701h when it is done.  If CF is set on
    return, PRINT will relinquish control immediately, and will not
    call AX=8701h
  PCVENUS (an early network shell by IBM and CMU) hooks this call to
    prevent background printing while its own code is active
SeeAlso: AH=80h,AH=81h,AX=8701h
--------P-2A8701-----------------------------
INT 2A CU - PRINT - END BACKGROUND PRINTING
  AX = 8701h
Desc: used to inform interested programs that PRINT has completed its
    background processing
Note: called by PRINT after it has performed some background printing; not
    called if AX=8700h returned with CF set.
SeeAlso: AX=8700h
--------N-2A89-------------------------------
INT 2A U - PC Network v1.00 - RECEIVER.COM - ???
  AH = 89h
  AL = ???  (ASSIGN uses 08h)
  ???
Return: ???
--------I-2A90-------------------------------
```

```
INT 2A U - IBM PC 3270 EMULATION PROGRAM - ???
  AH = 90h
  ???
Return: ???
Note: the LANtastic redirector and SERVER.EXE use this function with AL=01h,
    03h-07h,0Ch-11h
--------N-2AC2-------------------------------
INT 2A U - Network - ???
  AH = C2h
  AL = subfunction
       07h ???
       08h ???
  BX = 0001h
  ???
Return: ???
Note: this function is called by the DOS 3.30-6.00 APPEND
--------N-2AC4-------------------------------
INT 2A U - PC Network v1.00 - RECEIVER.COM - ???
  AH = C4h
  AL = subfunction
       07h ???
       08h ???
  BX = ???
  ???
Return: ???
SeeAlso: AH=86h
--------N-2AD800-----------------------------
INT 2A U - Novell NetWare Lite - SERVER - DOS CRITICAL SECTION DISABLE
  AX = D800h
Return: nothing
Desc: sets ??? flag, and sets ??? to initial value
Note: called by CLIENT for communication between client and server
SeeAlso: AX=D801h,AX=D850h
--------N-2AD801-----------------------------
INT 2A U - Novell NetWare Lite - SERVER - DOS CRITICAL SECTION ENABLE
  AX = D801h
Return: nothing
Desc: clears the ??? flag set by AX=D800h
Note: called by CLIENT for communication between client and server
SeeAlso: AX=D800h,AX=D850h
--------N-2AD850-----------------------------
```

```
INT 2A U - Novell NetWare Lite - CLIENT - START SERVER CRITICAL SECTION
  AX = D850h
Return: nothing
Desc: increments an internal byte-sized counter
Note: this function is intercepted by DV/X 1.10 PEERSERV.DVR and the
    Advanced NetWare 4.0 DOS Requester
SeeAlso: AX=D851h
--------N-2AD851-----------------------------
INT 2A U - Novell NetWare Lite - CLIENT - END SERVER CRITICAL SECTION
  AX = D851h
Return: nothing
Desc: resets an internal byte-sized counter to zero
Note: this function is intercepted by DV/X 1.10 PEERSERV.DVR and the
    Advanced NetWare 4.0 DOS Requester
SeeAlso: AX=D850h
--------N-2AD852-----------------------------
INT 2A U - Novell NetWare - DOS Requester v1.03 - SERVER LOADED
  AX = D852h
Return: ???
Note: calls the NetWare Lite SERVER installation check, and sets ??? pointer
SeeAlso: AX=D853h,INT 2F/AX=D880h
--------N-2AD853-----------------------------
INT 2A U - Novell NetWare - DOS Requester v1.03 - SERVER UNLOADED
  AX = D853h
Return: ???
Note: clears the pointer set by AX=D852h
SeeAlso: AX=D852h
--------N-2AE0------------------------------
INT 2A U - PC Network 1.00 - ???
  AH = E0h
  AL = subfunction??? (01h,02h, maybe others)
  ???
Return: ???
Note: called by PCNet 1.00 NET.COM, a shell program from which others are run
--------N-2AFF90----------------------------
INT 2A - PC/TCP PREDIR.EXE - ???
  AX = FF90h
Return: AX = ???
Note: PREDIR.EXE is the network printer redirector included as part of the
    PC/TCP system by FTP Software, Inc.
--------N-2AFF91----------------------------
```

```
INT 2A - PC/TCP PREDIR.EXE - ???
  AX = FF91h
  BX = ???
Return: AX = status???
--------N-2AFF92-----------------------------
INT 2A - PC/TCP PREDIR.EXE - INSTALLATION CHECK
  AX = FF92h
Return: AX = 0000h if installed
     BX = redirected printer port (FFFFh if no printers redirected)
     CX = version (CH = major, CL = minor)
Note: PREDIR.EXE is the network printer redirector included as part of the
    PC/TCP system by FTP Software, Inc.
--------N-2AFF93-----------------------------
INT 2A - PC/TCP PREDIR.EXE - ???
  AX = FF93h
Return: AX = ???
--------N-2AFF94-----------------------------
INT 2A - PC/TCP PREDIR.EXE - ???
  AX = FF94h
  BX = ???
  CX = ???
  DX = ???
Return: AX = ???
Note: PREDIR.EXE is the network printer redirector included as part of the
    PC/TCP system by FTP Software, Inc.
--------N-2AFF95-----------------------------
INT 2A - PC/TCP PREDIR.EXE - GET CONFIGURATION STRINGS
  AX = FF95h
  CX = what to get
      0000h ??? (returned pointer to "C:\COMMAND.COM")
      0001h spooling program
      0002h ???
      0003h spool file name
      0004h swap file name
Return: AX = status
      0000h successful
  BX:DX -> ASCIZ configuration string
--------N-2AFF96-----------------------------
INT 2A - PC/TCP PREDIR.EXE - SET PRINT JOB TERMINATION CONFIGURATION
  AX = FF96h
  CX = what to set
```

```
      0000h ???
      0001h print-on-hotkey state
      0002h print-on-exit state
      0003h print job timeout in clock ticks
      0004h print-on-EOF state
  BX = new value (0000h disabled, 0001h enabled except for timeout)
Return: AX = ???
SeeAlso: AX=FF97h
Note: PREDIR.EXE is the network printer redirector included as part of the
    PC/TCP system by FTP Software, Inc.
--------N-2AFF97-----------------------------
INT 2A - PC/TCP PREDIR.EXE - GET PRINT JOB TERMINATION CONFIGURATION
  AX = FF97h
  CX = what to get
      0000h ???
      0001h print-on-hotkey state
      0002h print-on-exit state
      0003h print job timeout in clock ticks
      0004h print-on-EOF state
Return: AX = status
      0000h successful
  BX = old value (0000h disabled, 0001 enabled except for timeout)
SeeAlso: AX=FF96h
--------D-2B---------------------------------
INT 2B - DOS 2+ - RESERVED
Note: this vector is not used in MS-DOS versions <= 6.22, and points at an
    IRET instruction
--------D-2B---------------------------------
INT 2B - IBM ROM-DOS v4.0 - ???
  AH = function
      00h ??? (modifies data in IBMBIO.COM)
      01h internal operations
      02h ???
    AL = index (00h-0Ch)
    Return: AX = ??? or (CMOS 2Dh and CMOS 2Eh)
      03h get ??? data
    Return: AX = (CMOS 2Dh and CMOS 2Eh)
      BX = FFFFh
      other does nothing
Note: function 03h is called by ROMSHELL.COM; if BX != 0, then the ES:DI from
    INT 2F/AX=1982h points at valid data
```

```
SeeAlso: INT 2F/AX=1982h
--------D-2C---------------------------------
INT 2C - DOS 2+ - RESERVED
Note: this vector is not used in DOS versions <= 6.00, and points at an IRET
--------O-2C---------------------------------
INT 2C - STARLITE architecture - KERNEL API
Note: STARLITE is an architecture by General Software for a series of MS-DOS
    compatible operating systems (OEM DOS, NETWORK DOS, and SMP DOS) to
    be released in 1991.  The interrupt number is subject to change
    before the actual release.
--------m-2C---------------------------------
INT 2C R - Cloaking - CALL PROTECTED-MODE PASSALONG CHAIN
Notes:  when this interrupt is invoked in V86 mode, RM386 will invoke the first
    in a chain of protected-mode handlers, and will only pass execution
    to the V86-mode INT 2C handler if none of the handlers in the
    passalong chain handle the call instead.  This is the method by which
    the real-mode stub of a cloaked application communicates with the
    protected-mode portion.
  the cloaking host calls the passalong chain with EAX=58494E33h ('WIN3')
    when MS Windows starts up and with EAX=334E4958h ('3NIW') when
    Windows shuts down; between these two broadcasts, the additional
    Windows-only Cloaking services are available
  this function was first introduced with RM386 (RAM-MAN/386) v6.00, the
    memory manager included in Helix Software's Netroom
SeeAlso: INT 2C/AX=0009h,INT 2F/AX=4310h"Cloaking"
--------m-2C0000-----------------------------
INT 2C P - Cloaking - ALLOCATE GDT SELECTOR
  AX = 0000h
  EBX = base address
  CL = access mode byte
  CH = extended access mode byte (omit limit field)
  EDX = segment limit
Return: CF clear if successful
      AX = selector
  CF set on error
      AX = error code (see #02556)
Notes:  this INT 2C interface is used by Netroom's DPMI.EXE v3.00
  to access extended memory, set the base address to the desired
    physical address plus 400000h (4M)
  this function was first introduced with RM386 (RAM-MAN/386) v6.00, the
    memory manager included in Helix Software's Netroom
```

```
SeeAlso: AX=0001h,AX=0002h,AX=0003h,AX=0004h,AX=0005h,INT 31/AH=57h,#00501

(Table 02556)
Values for Cloaking error code:
 0001h  no more selectors
 0002h  not a GDT ring 0 selector
 0003h  invalid selector (out of range, not user selector)
 0004h  selector not allocated
--------m-2C0001-----------------------------
INT 2C P - Cloaking - FREE GDT SELECTOR
  AX = 0001h
  SI = selector
Return: CF clear if successful
  CF set on error
      AX = error code (see #02556)
Note: this function was first introduced with RM386 (RAM-MAN/386) v6.00
SeeAlso: AX=0000h,INT 2F/AX=4310h"Cloaking"
--------m-2C0002-----------------------------
INT 2C P - Cloaking - SET SEGMENT BASE ADDRESS
  AX = 0002h
  SI = selector
  EBX = new physical base addres
Return: CF clear if successful
  CF set on error
      AX = error code (see #02556)
Note: this function was first introduced with RM386 (RAM-MAN/386) v6.00
SeeAlso: AX=0000h,AX=0003h,AX=0004h,INT 31/AX=0007h,#00501
--------m-2C0003-----------------------------
INT 2C P - Cloaking - SET SEGMENT LIMIT
  AX = 0003h
  SI = selector
  EBX = new limit
Return: CF clear if successful
  CF set on error
      AX = error code (see #02556)
Note: this function was first introduced with RM386 (RAM-MAN/386) v6.00
SeeAlso: AX=0000h,AX=0002h,AX=0004h,INT 31/AX=0008h
--------m-2C0004-----------------------------
INT 2C P - Cloaking - SET SEGMENT ACCESS MODE
  AX = 0004h
  SI = selector
```

```
  CL = new access mode byte (see #00502)
Return: CF clear if successful
  CF set on error
      AX = error code (see #02556)
Note: this function was first introduced with RM386 (RAM-MAN/386) v6.00
SeeAlso: AX=0000h,AX=0002h,AX=0003h,AX=0005h,INT 31/AX=0009h
--------m-2C0005-----------------------------
INT 2C P - Cloaking - SET SEGMENT EXTENDED ACCESS MODE
  AX = 0005h
  SI = selector
  CL = new extended access mode byte (limit field ignored) (see #02557)
Return: CF clear if successful
  CF set on error
      AX = error code (see #02556)
Note: this function was first introduced with RM386 (RAM-MAN/386) v6.00
SeeAlso: AX=0000h,AX=0002h,AX=0003h,AX=0004h,INT 31/AX=0009h

Bitfields for extended access mode byte:
Bit(s)  Description (Table 02557)
 7  4K granularity instead of byte granularity
 6  32-bit code segment
 5  reserved (0)
 4  segment available to system
SeeAlso: #00505
--------m-2C0006-----------------------------
INT 2C P - Cloaking - GET PROTECTED-MODE INTERRUPT VECTOR
  AX = 0006h
  CL = vector (00h-7Fh)
Return: CF clear
  DX:EBX -> current interrupt handler
Note: this function was first introduced with RM386 (RAM-MAN/386) v6.00
SeeAlso: AX=0007h,INT 31/AX=0204h
--------m-2C0007-----------------------------
INT 2C P - Cloaking - SET PROTECTED-MODE INTERRUPT VECTOR
  AX = 0007h
  CL = vector (00h-7Fh)
  DX:EBX -> interrupt handler
Return: CF clear
Notes:  this function was first introduced with RM386 (RAM-MAN/386) v6.00
  the IDT entry's type remains unchanged
SeeAlso: AX=0006h,INT 31/AX=0205h
```

```
--------m-2C0008-----------------------------
INT 2C P - Cloaking - GET PASSALONG ADDRESS
  AX = 0008h
Return: CF clear
  DX:EBX = current passalong address
Note: this function was first introduced with RM386 (RAM-MAN/386) v6.00
SeeAlso: AX=0009h,AX=002Ch,INT 2F/AX=4310h"Cloaking"
--------m-2C0009-----------------------------
INT 2C P - Cloaking - SET PASSALONG ADDRESS
  AX = 0009h
  DX:EBX = new value for passalong address (see #02558)
Return: CF clear
Notes:  when an INT 2C instruction is executed in V86 mode, the Cloaking host
    calls the passalong address.  The handler should check whether the
    upcall is of interest to it, and if not it should jump to the old
    passalong address (retrieved with AX=0008h before the handler was
    installed).  The final handler should return with CF clear to cause
    the interrupt to be reflected back to V86 mode if none of the
    passalong handlers is triggered
  this function was first introduced with RM386 (RAM-MAN/386) v6.00
SeeAlso: AX=0008h,AX=002Dh,INT 2C"PASSALONG CHAIN"

(Table 02558)
Values Cloaking passalong address is called with:
  EAX = CS:IP of byte following INT 2C instruction invoking passalong
  SS:EBX -> caller registers (see #02559)
  CF clear
  others undefined
Return: CF clear: pass along to V86-mode INT 2C handler
  CF set: return immediately to V86 mode

Format of Cloaking caller registers:
Offset  Size  Description (Table 02559)
 00h  DWORD EDI
 04h  DWORD ESI
 08h  DWORD EBP
 0Ch  DWORD reserved (ESP from PUSHAD instruction)
 10h  DWORD EBX
 14h  DWORD EDX
 18h  DWORD ECX
 1Ch  DWORD EAX
```

```
 20h  DWORD error code
 24h  DWORD EIP
 28h  WORD  CS
 2Ah  WORD  padding
 2Ch  DWORD EFLAGS
 30h  DWORD ESP
 34h  WORD  SS
 36h  WORD  padding
--remainder not available if protected-mode ring3 trap---
 38h  WORD  ES
 3Ah  WORD  padding
 3Ch  WORD  DS
 3Eh  WORD  padding
 40h  WORD  FS
 42h  WORD  padding
 44h  WORD  GS
 46h  WORD  padding
--------m-2C000A-----------------------------
INT 2C P - Cloaking - GET BASE ADDRESS OF GDT SELECTOR
  AX = 000Ah
  SI = selector
Return: CF clear if successful
      EBX = segment base address
  CF set on error
      AX = error code (see #02556)
Note: this function was first introduced with RM386 (RAM-MAN/386) v6.00, the
    memory manager included in Helix Software's Netroom
SeeAlso: AX=0000h,AX=0002h,AX=000Bh
--------m-2C000B-----------------------------
INT 2C P - Cloaking - GET SELECTOR LIMIT
  AX = 000Bh
  SI = selector
Return: CF clear if successful
      EBX = segment base address
  CF set on error
      AX = error code (see #02556)
Note: this function was first introduced with RM386 (RAM-MAN/386) v6.00
SeeAlso: AX=000Ah,INT 2F/AX=4310h"Cloaking"
--------m-2C---------------------------------
INT 2C P - RM386 v6.00 - CLOAKING - RESERVED FOR CLOAKED BIOS USE UNDER WINDOWS
  AX = function (000Ch-001Fh)
```

```
--------m-2C000F-----------------------------
INT 2C P - Cloaking v1.01 - "Simulate_Shell_Event"
  AX = 000Fh
  ECX = event code (see #02560)
  DX = subfunction for event
  EDX high word = boost value (see #02561)
  SI:EDI -> completion procedure
Return: CF clear if successful (event scheduled)
  CF set on error
Note: this function is only available while MS Windows is running
SeeAlso: AX=0011h,AX=0012h,INT 2F/AX=1605h,INT 2F/AX=4310h"Cloaking"

(Table 02560)
Values for Cloaking shell event code:
 0414h  Hot key event
  subevent 0000h: Alt-Space
  subevent 0001h: Alt-Enter
  subevent 0002h: Dir-VM
 0415h  Switch context
  subevent 0000h for DOS VM context, nonzero for System VM context
 0416h  Clipboard event
 0417h  Termination event
  subevent 0000h for normal termination, nonzero for error
 0418h  Display message
  subevent 0000h for normal message, nonzero for system model ASAP
 0419h  Crash
 041Ah  Paste complete
  subevent 0000h: normal
  subevent 0001h: cancelled by user
  subevent 0002h: cancelled
 041Bh  Contention event
 041Ch  Screen switch
  subevent 0000h: forward
  subevent 0001h: back
 041Dh  Filesystem change
 041Eh  Check Focus
 041Fh  Panic

Bitfields for boost value:
Bit(s)  Description (Table 02561)
 0  boost system VM until focus changes
```

```
 1  boost system VM on Switcher screen
 2  boost system VM until response
 3  boost system VM during clipboard activity
 4  boost system VM during print screen
 5  boost system VM during update
--------m-2C0011-----------------------------
INT 2C P - Cloaking v1.01 - "Switch_VMs_and_Call_back"
  AX = 0011h
  EBX = handle of VM to be made active
  SI:EDI -> 32-bit FAR completion procedure
Return: CF clear if successful (scheduled)
  CF set on error
Notes:  this function is only available while MS Windows is running
  the completion procedure is called with CF clear if the specified
    VM has been made active, or with CF set on error
SeeAlso: AX=000Fh,AX=0012h
--------m-2C0012-----------------------------
INT 2C P - Cloaking v1.01 - "Query_Current_VM"
  AX = 0012h
Return: CF clear
  EBX = handle of active VM
  ESI = handle of system VM
  ECX = VM status flags (see #02562)
  EDX = shell flags (see #02563)
Note: this function is only available while MS Windows is running in enhanced
    mode
SeeAlso: AX=000Fh,AX=0011h,AX=0013h

Bitfields for VM status flags:
Bit(s)  Description (Table 02562)
 0  in exclusive mode
 1  runs in background
 2  being created
 3  suspended
 4  not executable
 5  executing in protected mode
 6  contains PM application
 7  32-bit PM application
 8  called from VxD
 9  high priority background
 10 blocked on semaphore
```

```
 11 awakening
 12 has pageable V86
 13 has locked V86
 14 is scheduled
 15 idle
 16 closing

Bitfields for shell flags:
Bit(s)  Description (Table 02563)
 2  windowed
 5  Alt-Tab reserved
 6  Alt-Esc reserved
 7  Alt-Space reserved
 8  Alt-PrtSc reserved
 9  Alt-Enter reserved
 10 Alt-PrtSc reserved
 11 PrtSc reserved
 12 polling enabled
 13 no HMA
 14 has shortcut key
 15 locked EMS handles
 16 locked XMS handles
 17 fast paste enabled
 18 locked V86 memory
 30 close-on-exit enabled
--------m-2C0013-----------------------------
INT 2C P - Cloaking v1.01 - "Issue_System_Modal_Message"
  AX = 0013h
  EDX = message box flags (see #02564)
  DS:ECX -> ASCIZ message text
  DS:EDI -> ASCIZ caption
Return: CF clear
  EAX = response code
Note: this function is only available while MS Windows is running in enhanced
    mode
SeeAlso: AX=000Fh,AX=0012h

Bitfields for message box flags:
Bit(s)  Description (Table 02564)
 3-0  response codes (see #02565)
 7-4  icon codes
```

```
  1 = Warning hand
  2 = exclamation mark
  4 = asterisk
 9-8  default response (0 = first button, 1 = second, 2 = third)
 12 message is system model
 15 don't change focus
 29 hang with interrupts enabled
 30 do not window
 31 execute ASAP

(Table 02565)
Values for response codes:
 00h  OK
 01h  OK, Cancel
 02h  Abort, Retry, Ignore
 03h  Yes, No, Cancel
 04h  Yes, No
 05h  Retry, Cancel
--------m-2C001D-----------------------------
INT 2C P - Cloaking v1.01 - GET INT 2C API HANDLER ENTRY POINT
  AX = 001Dh
Return: CF clear
  DX:EBX = selector:offset of Cloaking host INT 2C handler
Desc: get the Cloaking host's entry point to bypass any other programs
    which may have hooked INT 2C in protected mode
Note: the returned entry point must be called with a simulated INT, i.e.
    a PUSHD must precede the far call to the handler
SeeAlso: INT 2F/AX=4310h"Cloaking"
--------m-2C001E-----------------------------
INT 2C P - Cloaking v1.01 - CLEAR CRITICAL SECTION
  AX = 001Eh
Return: CF clear
Desc: allow MS Windows to switch to another VM after having prevented it
    by invoking a critical section
SeeAlso: AX=001Fh,INT 15/AX=101Ch,INT 2F/AX=1682h
--------m-2C001F-----------------------------
INT 2C P - Cloaking v1.01 - SET CRITICAL SECTION
  AX = 001Fh
Return: CF clear
Desc: prevent MS Windows from switching to another VM
SeeAlso: AX=001Eh,INT 15/AX=101Bh,INT 2F/AX=1681h
```

```
--------m-2C0020------------------------------
INT 2C P - Cloaking - GET SIZE OF PROTECTED-MODE STATE
  AX = 0020h
Return: EAX = number of bytes required for storing state
Note: this function was first introduced with RM386 (RAM-MAN/386) v6.00
SeeAlso: AX=0021h,AX=0022h
--------m-2C0021------------------------------
INT 2C P - Cloaking - SAVE PROTECTED-MODE STATE
  AX = 0021h
  ES:EDI -> buffer for protected-mode state
Return: CF clear
  buffer filled
Note: this function was first introduced with RM386 (RAM-MAN/386) v6.00
SeeAlso: AX=0020h,AX=0022h
--------m-2C0022------------------------------
INT 2C P - Cloaking - RESTORE PROTECTED-MODE STATE
  AX = 0022h
  DS:ESI -> buffer containing previously-saved protected-mode state
Return: CF clear if successful
      state restored
  CF set on error (invalid buffer contents)
Note: this function was first introduced with RM386 (RAM-MAN/386) v6.00
SeeAlso: AX=0020h,AX=0021h
--------m-2C0023------------------------------
INT 2C P - Cloaking - ISSUE PROTECTED-MODE XMS CALL
  AX = 0023h
Notes:  not currently implemented--NOP in RM386 v6.00
  this function was first introduced with RM386 (RAM-MAN/386) v6.00
--------m-2C0024------------------------------
INT 2C P - Cloaking - SET V86-MODE STACK
  AX = 0024h
  DX:EBX = new value for V86-mode SS:ESP
Return: nothing
Note: this function was first introduced with RM386 (RAM-MAN/386) v6.00
--------m-2C0025------------------------------
INT 2C P - Cloaking - CALL V86-MODE PROCEDURE
  AX = 0025h
  DS:EBX -> client register structure (see #02559)
Return: CF clear if successful
      client register structure updated
  CF set if no more nested procedure call space available
```

```
Notes:  this call uses the V86-mode stack supplied in the client structure, and
    calls the routine specified by CS:IP in the client structure
  this function was first introduced with RM386 (RAM-MAN/386) v6.00
SeeAlso: AX=0026h,AX=0027h,INT 31/AX=0301h
--------m-2C0026-----------------------------
INT 2C P - Cloaking - CALL V86-MODE INTERRUPT HANDLER
  AX = 0026h
  DS:EBX -> client register structure (see #02559)
  CX = interrupt number
Return: CF clear if successful
      client register structure updated
  CF set if no more nested procedure call space available
Notes:  this call uses the V86-mode stack supplied in the client structure
  this function was first introduced with RM386 (RAM-MAN/386) v6.00
SeeAlso: AX=0025h,AX=0027h,INT 31/AX=0300h
--------m-2C0027-----------------------------
INT 2C P - Cloaking - CHAIN TO V86-MODE INTERRUPT HANDLER
  AX = 0027h
  DS:EBX -> client register structure (see #02559)
Return: CF clear if successful
      client register structure updated
  CF set if no more nested procedure call space available
Notes:  this call uses the V86-mode stack supplied in the client structure,
    and jumps to the address specified by CS:IP in the client structure
  this function was first introduced with RM386 (RAM-MAN/386) v6.00
SeeAlso: AX=0025h,AX=0026h
--------m-2C0028-----------------------------
INT 2C P - Cloaking - GET ESP0 FROM TSS
  AX = 0028h
Return: CF clear
  EAX = TSS's ESP0
Note: this function was first introduced with RM386 (RAM-MAN/386) v6.00, the
    memory manager included in Helix Software's Netroom
--------m-2C0029-----------------------------
INT 2C P - Cloaking - SET SECONDARY STACK
  AX = 0029h
  DX:EBX = new value for SS:ESP of ring 3 secondary stack
Return: CF clear
Desc: inform RM386 of the ring 3 interrupt stack location
Note: this function was first introduced with RM386 (RAM-MAN/386) v6.00
--------m-2C002A-----------------------------
```

```
INT 2C P - Cloaking - SET 8259 IRQ BASE VECTORS
  AX = 002Ah
  BL = base vector of master interrupt controller
  CL = base vector of slave interrupt controller
Notes:  this call merely informs RM386 that the caller has changed the
    interrupt mappings
  this function was first introduced with RM386 (RAM-MAN/386) v6.00
SeeAlso: INT 67/AX=DE0Bh
--------m-2C002BCH81-------------------------
INT 2C P - Cloaking - PROTECTED-MODE VIRTUAL DMA SERVICES
  AX = 002Bh
  CH = 81h
  CL = subfunction (02h-0Ch)
  other registers as appropriate for subfunction
Return: varies by function
  CF set on error
Notes:  these functions are equivalent to the INT 4B/AX=81xxh subfunctions
    with the same numbers
  this function was first introduced with RM386 (RAM-MAN/386) v6.00
SeeAlso: INT 4B/AX=8102h,INT 4B/AX=810Ch
--------m-2C002C-----------------------------
INT 2C P - Cloaking - GET PORT-TRAPPING PASSALONG
  AX = 002Ch
Return: CF clear
  DX:EBX = current I/O trapping passalong address
Note: this function was first introduced with RM386 (RAM-MAN/386) v6.00
SeeAlso: AX=0008h,AX=002Dh
--------m-2C002D-----------------------------
INT 2C P - Cloaking - SET PORT-TRAPPING PASSALONG
  AX = 002Dh
  DX:EBX = new I/O trapping passalong address (see #02566)
Return: CF clear
Notes:  RM386 calls the passalong address whenever an access to a monitored
    I/O port is attempted; the handler should check whether it is a port
    that it is interested in, and if not call the previous passalong
    address (which was retrieved with AX=002Ch before installing the
    new handler)
  this function was first introduced with RM386 (RAM-MAN/386) v6.00
SeeAlso: AX=0009h,AX=002Ch,INT 67/AX=5DEAh

(Table 02566)
```

```
Values Cloaking port-trapping passalong address is called with:
  EAX = CS:IP of faulting instruction (unless executing in protected-mode
    ring 3)
  SS:EBX -> caller register structure (see #02559)
    check EFLAGS V86-mode bit for type
  CX = first two bytes of I/O instruction which was trapped
  DX = port to which I/O is being performed
  CF clear
Return: CF clear if RM386 should perform I/O operation
  CF set if I/O should be skipped
Note: RM386 skips the trapped I/O instruction, so the passalong handler
    should not modify the client CS:EIP
--------m-2C002E-----------------------------
INT 2C P - Cloaking - TRAP I/O PORT
  AX = 002Eh
  DX = port number to trap
Return: CF clear if successful
  CF set on error (port out of range or reserved)
Note: this function was first introduced with RM386 (RAM-MAN/386) v6.00
SeeAlso: AX=002Fh,AX=0030h
--------m-2C002F-----------------------------
INT 2C PU - Cloaking - UNTRAP I/O PORT
  AX = 002Fh
  DX = port number for which to cancel trapping
Return: CF clear if successful
  CF set on error (port out of range or reserved)
Note: this function was first introduced with RM386 (RAM-MAN/386) v6.00
SeeAlso: AX=002Eh,AX=0030h
--------m-2C0030-----------------------------
INT 2C PU - Cloaking - GET TRAPPING STATE OF SPECIFIED PORT
  AX = 0030h
  DX = port number
Return: CF clear if successful
     BX = current state (0000h not trapped, 0001h trapped)
  CF set on error (port out of range or reserved)
Note: this function was first introduced with RM386 (RAM-MAN/386) v6.00
SeeAlso: AX=002Eh,AX=002Fh
--------m-2C0031-----------------------------
INT 2C PU - RM386 v6.00 - BUG
  AX = 0031h
Program: RM386 (RAM-MAN/386) is the memory manager included in Helix
```

```
    Software's Netroom
Note: due to a fencepost error, RM386 v6.00 will branch unpredictably if
    invoked with this function
--------m-2C0031-----------------------------
INT 2C P - Cloaking v1.01 - ALLOCATE V86 CALLBACK
  AX = 0031h
  DX:EBX = CS:EIP of protected-mode routine to be invoked by callback
Return: CF clear if successful
      EBX = CS:IP of V86-mode callback handler
  CF set on error
SeeAlso: AX=0032h
--------m-2C0032-----------------------------
INT 2C P - Cloaking v1.01 - FREE V86 CALLBACK
  AX = 0032h
  EBX = CS:IP of V86-mode callback handler
Return: CF clear if successful
  CF set on error
      AX = error code
    0005h invalid callback address
    0006h callback already free
SeeAlso: AX=0032h
--------m-2C0033-----------------------------
INT 2C P - Cloaking v1.01 - REGISTER CLOAKING CLIENT
  AX = 0033h
  DS:EDX -> client registration structure (see #02567)
Return: CF clear if successful
  CF set on error (linked list corrupt)
SeeAlso: AX=0034h,#02778 at INT 2F/AX=4310h"Cloaking"

Format of client registration structure:
Offset  Size  Description (Table 02567)
 00h  PWORD link to next structure
 06h  PWORD link to previous structure
 0Ch  2 BYTEs client version (major, minor)
 0Eh 20 BYTEs client name
 22h  DWORD physical address of client start
 26h  DWORD client's total size in bytes
Note: the link area should not be modified once the structure has been
    used for the registration call
--------m-2C0034-----------------------------
INT 2C P - Cloaking v1.01 - UNREGISTER CLOAKING CLIENT
```

```
  AX = 0034h
  DS:EDX -> client registration structure (see #02567)
Return: CF clear if successful
  CF set on error (linked list corrupt)
Note: the client must unregister before freeing the XMS block containing
    its registration structure(s)
SeeAlso: AX=0033h,#02778 at INT 2F/AX=4310h"Cloaking"
--------D-2D---------------------------------
INT 2D - DOS 2+ - RESERVED
Note: this vector is not used in DOS versions <= 6.00, and points at an IRET
BUG:  RM386 v6.00-6.02 (as distributed with Helix's Netroom v3.x) contains
    a stack bug in its protected-mode INT 2D handler which causes a crash
    when INT 2D is invoked from V86 mode
--------t-2D---------------------------------
INT 2D - ALTERNATE MULTIPLEX INTERRUPT SPECIFICATION (AMIS) [v3.6]
  AH = multiplex number
  AL = function
       00h installation check
       01h get private entry point
       02h uninstall
       03h request popup
       04h determine chained interrupts
       05h get hotkey list
       06h get device-driver information
       07h-0Fh reserved for future enhancements
    Return: AL = 00h (not implemented)
      other  application-dependent
  other registers vary by function (also see individual entries below)
Return: varies by function
Notes:  programs should not use fixed multiplex numbers; rather, a program
    should scan all multiplex numbers from 00h to FFh, remembering the
    first unused multiplex in case the program is not yet installed.
    For multiplex numbers which are in use, the program should compare
    the first 16 bytes of the signature string to determine whether it
    is already installed on that multiplex number.  If not previously
    installed, it should use the first free multiplex number.
  functions other than 00h are not valid unless a program is installed
    on the selected multiplex number
  to be considered fully compliant with version 3.6 of the specification,
    programs must implement at least functions 00h, 02h (no resident
    uninstall code required), and 04h (return value 04h).  TSRs that
```

```
    provide hotkeys with which the user can activate them must also
    implement function 05h.  TSRs which provide DOS device drivers must
    also implement function 06h.  The absolute minimum fully-compliant
    implementation has an overhead of 64 bytes (80 bytes with function
    05h) plus 22 bytes per hooked interrupt (for the interrupt sharing
    protocol header and hook list entry).
  the signature string and description may be used by memory mappers
    to display the installed programs
  to be considered fully compliant, users of this specification must
    adhere to the IBM interrupt sharing protocol (see #02568), which will
    permit removal of TSRs in arbitrary order and interrupt handler
    reordering.  All TSRs following this specification should be
    removable unless they are loaded from CONFIG.SYS, though they need
    not keep the code for removing themselves resident; it is acceptable
    for a separate program to perform the interrupt unhooking and
    memory-freeing steps of removal.
  A sample public-domain implementation including example TSRs and
    utility programs may be found in a separate package distributed as
    AMISLnnn.ZIP (AMISL092.ZIP as of this writing).
  Please let me know if you choose to follow this proposal.  The
    signature and a list of the private API calls you use would be
    appreciated, as well.
SeeAlso: INT 2D/AL=00h,INT 2D/AL=01h,INT 2D/AL=02h,INT 2D/AL=03h,INT 2D/AL=04h
SeeAlso: INT 2D/AL=05h,INT 2D/AL=06h,INT 2F"NOTES"

Format of interrupt sharing protocol interrupt handler entry point:
Offset  Size  Description (Table 02568)
 00h  2 BYTEs short jump to actual start of interrupt handler, immediately
      following this data block (EBh 10h)
 02h  DWORD address of next handler in chain
 06h  WORD  signature 424Bh
 08h  BYTE  EOI flag
    00h software interrupt or secondary hardware interrupt handler
    80h primary hardware interrupt handler (will issue EOI to
          interrupt controller)
 09h  2 BYTEs short jump to hardware reset routine
    must point at a valid FAR procedure (may be just RETF)
 0Bh  7 BYTEs reserved (0) by IBM for future expansion
Note: when chaining to the prior handler, the interrupt handler must perform
    an indirect jump/call using the address at offset 02h in the
    ISP header.  This permits another AMIS TSR to hook itself into
```

```
    the chain at a position other than as the first handler to receive
    an interrupt.
SeeAlso: INT F1/AH=01h"Common ISDN API",INT F1/AH=06h"CAPI",#04068
--------t-2D--00----------------------------
INT 2D - AMIS v3.0+ - INSTALLATION CHECK
  AL = 00h
  AH = multiplex number for program
Return: AL = 00h if free
  AL = FFh if multiplex number in use
      CX = binary version number (CH = major, CL = minor)
      DX:DI -> signature string (see #02569) identifying the program
      using the multiplex number
SeeAlso: INT 2D/AL=01h,INT 2D/AL=02h,INT 2D/AL=03h,INT 2D/AL=04h,INT 2D/AL=05h
SeeAlso: INT 2D/AL=06h
Index:  installation check;Alternate Multiplex Interrupt Specification
Index:  installation check;AMIS|installation check;FASTMOUS
Index:  installation check;SPELLER|installation check;Monitor
Index:  installation check;NOLPT|installation check;NOTE
Index:  installation check;RBkeyswp|installation check;SWITCHAR
Index:  installation check;VGABLANK|installation check;EATMEM
Index:  installation check;RECALL|installation check;XPTR2

Format of AMIS signature string:
Offset  Size  Description (Table 02569)
 00h   8 BYTEs blank-padded manufacturer's name (possibly abbreviated)
 08h   8 BYTEs blank-padded product name
 10h 64 BYTEs ASCIZ product description (optional, may be a single 00h)
Note: it is not necessary to reserve a full 64 bytes for the description,
    just enough to store the actual ASCIZ string
SeeAlso: #02570

(Table 02570)
Values for AMIS signatures known to be in use:
 'Byrial J' 'EKLAVO  '  permits keyboard entry of Esperanto accented letters
 'CoveSoft' 'Burnout+'  shareware screen saver Burnout Plus
 'Crynwr  ' 'SPELLER '  TSR spelling-checker
 'CPH1995 ' 'CDTSR   '  resident CD-Audio player
 'CPH1996 ' 'DSAPI   '
 'CSJewell' 'Modula3L'  Curtis Jewell's Modula-3 compiler (non-TSR)
 'Cubbi...' 'ASCII...'  Example ASCII code view/enter TSR by Sergey Zubkov
 'Cubbi...' 'ScrnGrab'  Example screen grabber by Sergey Zubkov
```

```
'DAISYCHA' 'INDRIVER'  Advanced Parallel Port daisy chain driver (vendor name
       in product description field, if desired)
     (see also INT 2D/AL=DCh)
'DTown SD' 'DTU     '  DTown Software Development's DTown Utilities
     (see also INT 2D/AL=20h)
'ECLIPSE ' 'PLUMP   '  Eclipse Software's printer and plotter spooler
'GraySoft' 'GIPC    '  GraySoft's Inter-Process Communications driver
'heathh  ' 'Monitor '
'Helge O '   TSRs by Helge Olav Helgesen
'IVALM SK' 'lmkey   '  Russian keyboard driver by Sergey Khabarov
'IVALM SK' 'lmrus   '  Russian screen   driver by Sergey Khabarov
'J. Berry' 'RATSR   '  RemoteAccess Network Manager workstation module
'JWB   ' 'RAMLIGHT'  James Birdsall's on-screen RAMdisk activity indicator
'M Better' 'iHPFS   '  Marcus Better's HPFS filesystem driver for DOS
'M. Paul ' 'FREEVER '  DOS version-faking TSR by Matthias Paul
'Nildram ' 'ST      '  Screen Thief graphics screen grabber
'NoBrain ' 'FlatReal'  Flat real mode monitor by Sergei Shtylyov
'NoBrain ' 'Grabber '  Frame grabber drivers by Sergei Shtylyov
'NoBrain ' 'SqrModes'  TSR providing some non-standard video modes
     by Sergei Shtylyov
'Pino Nav' 'ALTMENU '  activate any program's menu bar by pressing Alt key
'Pino Nav' 'Keybit  '  Pino Navato's KEYBIT Lite Italian keyboard driver v4+
'PowrQuot' 'CAPRILOG'
'PowrQuot' 'CAPRITSR'
'PowrQuot' 'CAPRIWIN'
'R-Ware  ' 'dLite   '  run-time data decompression TSR
'Ralf B  ' 'disaXXYY'  RBdisabl -- disable key scancode XX w/ shift states YY
'Ralf B  ' 'DUALVGA '  dual-VGA support, screen blanker, and DPMS driver
'Ralf B  ' 'FASTMOUS'  example TSR included with sample AMIS library code
'Ralf B  ' 'NoBreak '  disable Ctrl-@, Ctrl-C, and Ctrl-Break keys
'Ralf B  ' 'NOLPT n '  example TSR -- turn LPTn into bit-bucket
'Ralf B  ' 'NOTE    '  example TSR -- popup note-taker
'Ralf B  ' 'RBclock '  RBclock -- on-screen real-time clock
'Ralf B  ' 'RBclockE'  RBclock -- on-screen elapsed-time clock
'Ralf B  ' 'RBdvorak'  Dvorak keyboard mapping w/ opt Esc/~, LCtrl/CapsLk swap
'Ralf B  ' 'RBkcount'  display count of keystrokes on screen
'Ralf B  ' 'RBkeyswp'  RBkeyswap v3.0+ -- swap Esc/~ and LCtrl/CapsLock keys
'Ralf B  ' 'RBnoboot'  disable Ctrl-Alt-Del key combination
'Ralf B  ' 'ShftCaps'  require Shift-CapsLock to turn on CapsLock
'Ralf B  ' 'ShftNumL'  require Shift-NumLock to turn off NumLock
'Ralf B  ' 'SWITCHAR'  example TSR -- add switchar() support removed from DOS5
```

```
 'Ralf B   ' 'VGABLANK'  VGA-only screen blanker
 'Ralf B   ' 'WINTAME '  yield CPU when program in Win95 DOS box is idle
 'Sally IS' 'Mdisk   '  removeable, resizeable RAMdisk
 'Sally IS' 'Scr2Tex '  screen dumper with output in (La)Tex format
 'SRT    ' 'STOPBOOT'  reboot preventer by Steve Talbot
 'Thaco    ' 'NEST    '  Eirik Pedersen's programmer's delimiter matcher
 'TifaWARE' 'EATMEM  '  George A. Theall's public domain memory restrictor for
      testing programs (v1.1+)
 'TifaWARE' 'RECALL  '  public domain commandline editor and history (v1.2+)
 'Todd     ' 'XPTR2   '  PC-to-Transputer interface by Todd Radel
 'WlkngOwl' 'NoiseSYS'  NOISE.SYS random-number generator
SeeAlso: #02569
--------t-2D--01-----------------------------
INT 2D - AMIS v3.0+ - GET PRIVATE ENTRY POINT
  AL = 01h
  AH = multiplex number for program
Return: AL = 00h if all API calls via INT 2D
  AL = FFh if entry point supported
      DX:BX -> entry point for bypassing interrupt chain
Note: this function is not valid unless a program is installed on the
    specified multiplex number; use INT 2D/AL=00h to check
SeeAlso: INT 2D/AL=00h,INT 2D/AL=02h,INT 2D/AL=03h,INT 2D/AL=04h,INT 2D/AL=05h
SeeAlso: INT 2D/AL=06h
Index:  entry point;Alternate Multiplex Interrupt|entry point;AMIS
--------t-2D--02-----------------------------
INT 2D - AMIS v3.0+ - UNINSTALL
  AL = 02h
  AH = multiplex number for program
  DX:BX = return address for successful uninstall (may be ignored by TSR)
Return: AL = status
      00h not implemented (makes TSR non-compliant with specification)
      01h unsuccessful
      02h can not uninstall yet, will do so when able
      03h safe to remove, but no resident uninstaller
      (TSR still enabled)
    BX = segment of memory block with resident code
      04h safe to remove, but no resident uninstaller
      (TSR now disabled)
    BX = segment of memory block with resident code
      05h not safe to remove now, try again later
      06h disabled, but can not be removed from memory
```

```
      because loaded from CONFIG.SYS
      07h safe to remove, but no resident device-driver
      uninstaller.  Caller must unlink device
      drivers from DOS device chain as well as
      unhooking interrupts and freeing memory
    BX = segment of memory block with resident code
      FFh successful
  return at DX:BX with AX destroyed if successful and TSR honors
    specific return address
Note: this function is not valid unless a program is installed on the
    specified multiplex number; use INT 2D/AL=00h to check
SeeAlso: INT 2D/AL=00h,INT 2D/AL=01h,INT 2D/AL=03h,INT 2D/AL=04h,INT 2D/AL=05h
SeeAlso: INT 2D/AL=06h
Index:  uninstall;Alternate Multiplex Interrupt Specification|uninstall;AMIS
--------t-2D--03-----------------------------
INT 2D - AMIS v3.0+ - REQUEST POP-UP
  AL = 03h
  AH = multiplex number for program
Return: AL = status
      00h not implemented or TSR is not a pop-up
      01h can not pop up at this time, try again later
      02h can not pop up yet, will do so when able
      03h already popped up
      04h unable to pop up, user intervention required
    BX = standard reason code
        0000h unknown failure
        0001h interrupt chain passes through memory
        which must be swapped out to pop up
        0002h swap-in failed
    CX = application's reason code if nonzero
      FFh TSR popped up and was exited by user
    BX = return value
        0000h no return value
        0001h TSR unloaded
        0002h-00FFh reserved
        0100h-FFFFh application-dependent
Note: this function is not valid unless a program is installed on the
    specified multiplex number; use INT 2D/AL=00h to check
SeeAlso: INT 2D/AL=00h,INT 2D/AL=01h,INT 2D/AL=02h,INT 2D/AL=04h,INT 2D/AL=05h
SeeAlso: INT 2D/AL=06h
--------t-2D--04-----------------------------
```

```
INT 2D - AMIS v3.0+ - DETERMINE CHAINED INTERRUPTS
  AL = 04h
  AH = multiplex number for program
  BL = interrupt number (except 2Dh)
Return: AL = status
      00h not implemented (makes TSR non-compliant with specification)
      01h (obsolete) unable to determine
      02h (obsolete) interrupt hooked
      03h (obsolete) interrupt hooked, address returned
    DX:BX -> TSR's interrupt BL handler
      04h list of hooked interrupts returned
    DX:BX -> interrupt hook list (see #02571)
      FFh interrupt not hooked
Notes:  BL is ignored if the TSR returns AL=04h; in that case, the caller
    needs to scan the return list rather than making additional calls
    to this function.  If the return is not 00h or 04h, then the caller
    must cycle through the remaining interrupt numbers it wishes to
    check.
  return values 01h through 03h may not be used by AMIS v3.6-compliant
    programs; they are included here solely for compatibility with
    version 3.3, though they were probably never used in any
    implementation
  for return values 01h through 03h, since INT 2D is known to be hooked,
    the resident code need not test for BL=2Dh (to minimize its size),
    and the return value is therefore undefined in that case.
  this function is not valid unless a program is installed on the
    specified multiplex number; use INT 2D/AL=00h to check
SeeAlso: INT 2D/AL=00h,INT 2D/AL=01h,INT 2D/AL=02h,INT 2D/AL=03h,INT 2D/AL=05h
SeeAlso: INT 2D/AL=06h

Format of AMIS interrupt hook list [array]:
Offset  Size  Description (Table 02571)
 00h  BYTE  interrupt number (last entry in array is 2Dh)
 01h  WORD  offset within hook list's segment of the interrupt handler
    this will point at the initial short jump of the interrupt
      sharing protocol header (see #02568)
SeeAlso: #02572
--------t-2D--05----------------------------
INT 2D - AMIS v3.5+ - GET HOTKEYS
  AL = 05h
  AH = multiplex number for program
```

```
Return: AL = status
      00h not implemented
      FFh supported
    DX:BX -> hotkey list (see #02572)
Notes:  this function is not valid unless a program is installed on the
    specified multiplex number; use INT 2D/AL=00h to check
  programs which provide hotkeys are required to provide this function
    to be fully compliant with this specification
SeeAlso: INT 2D/AL=00h,INT 2D/AL=01h,INT 2D/AL=02h,INT 2D/AL=03h,INT 2D/AL=04h
SeeAlso: INT 2D/AL=06h

Format of AMIS hotkey list:
Offset  Size  Description (Table 02572)
 00h  BYTE  type of hotkey checking (see #02573)
 01h  BYTE  number of hotkeys (may be zero if TSR can disable hotkeys)
 02h 6N BYTEs array of hotkey definitions
    (one per hotkey, first should be primary hotkey)
    Offset  Size  Description
     00h  BYTE  hotkey scan code (00h/80h if shift states only)
        hotkey triggers on release if bit 7 set
     01h  WORD  required shift states (see #02574)
     03h  WORD  disallowed shift states (see #02574)
     05h  BYTE  hotkey flags (see #02575)
Notes:  except for bit 7, the shift states correspond exactly to the return
    values from INT 16/AH=12h.  A set bit in the required states word
    indicates that the corresponding shift state must be active when the
    hotkey's scan code is received for the hotkey to be recognized; a
    clear bit means that the corresponding state may be ignored.  A set
    bit in the disallowed shift states word indicates that the
    corresponding shift state must be inactive.
  for the disallowed-states word, if one of the "either" bits is set,
    then both the corresponding left bit and right bit must be set
  examples:
    Ctrl-Alt-Del monitoring: 53h 000Ch 0003h 06h
    Alt-key tap (DESQview):  B8h 0000h 0007h 08h
    Shf-Shf-N (NOTE.COM):  31h 0003h 000Ch 00h
Index:  hotkeys;AMIS
SeeAlso: #00006

Bitfields for type of AMIS hotkey checking:
Bit(s)  Description (Table 02573)
```

```
 0  checks before chaining INT 09
 1  checks after chaining INT 09
 2  checks before chaining INT 15/AH=4Fh
 3  checks after chaining INT 15/AH=4Fh
 4  checks on INT 16/AH=00h,01h,02h
 5  checks on INT 16/AH=10h,11h,12h
 6  checks on INT 16/AH=20h,21h,22h
 7  reserved (0)
SeeAlso: #02572

Bitfields for AMIS shift states:
Bit(s)  Description (Table 02574)
 0  right shift pressed
 1  left shift pressed
 2  either control key pressed
 3  either Alt key pressed
 4  ScrollLock active
 5  NumLock active
 6  CapsLock active
 7  either shift key pressed
 8  left control key pressed
 9  left Alt key pressed
 10 right control key pressed
 11 right Alt key pressed
 12 ScrollLock pressed
 13 NumLock pressed
 14 CapsLock pressed
 15 SysReq key pressed
Notes:  if bit 2 is set, either control key may be pressed for the hotkey; if
    bits 8 and 10 are both set, then both control keys must be pressed.
    Similarly for bits 3 and 9/11, as well as 7 and 0/1.
  the SysReq key is often labeled SysRq
SeeAlso: #02572,#02575

Bitfields for AMIS hotkey flags:
Bit(s)  Description (Table 02575)
 0  hotkey chained before processing
 1  hotkey chained after processing
 2  others should pass through this hotkey so that it can be monitored
 3  hotkey will not activate if other keys pressed/released before hotkey
    press is completed
```

```
 4  this key is remapped into some other key
 5  this key is conditionally chained (sometimes passed on, sometimes
    swallowed)
 6-7  reserved (0)
SeeAlso: #02572,#02574
--------t-2D--06-----------------------------
INT 2D - AMIS v3.6 - GET DEVICE-DRIVER INFORMATION
  AL = 06h
  AH = multiplex number for program
Return: AL = number of device driver headers supplied by prog.
  AH = device-driver flags (see #02576)
  DX:BX -> first device driver header (see #01646)
Program: AMIS is the Alternate Multiplex Interrupt Specification promulgated
    by Ralf Brown
Notes:  if AL=00h, AH,BX,DX are meaningless and may be destroyed
  this function is not valid unless a program is installed on the
    specified multiplex number; use INT 2D/AL=00h to check
  programs which provide device drivers are required to support this
    function to be considered fully compliant with v3.6+ of the
    specification
SeeAlso: INT 2D/AL=00h,INT 2D/AL=01h,INT 2D/AL=02h,INT 2D/AL=03h,INT 2D/AL=04h
SeeAlso: INT 2D/AL=05h

Bitfields for AMIS device-driver information flags:
Bit(s)  Description (Table 02576)
 0  program loaded from CONFIG.SYS, and thus can not be removed from memory
  (leave clear if unable to determine)
 1  device driver headers have not been linked into DOS device chain
 2  reentrant device driver(s)
--------N-2D--10-----------------------------
INT 2D - RATSR 2.0+ - GET STATUS
  AL = 10h
  AH = AMIS multiplex number for RATSR
Return: AL = status
      01h listening (no connection)
      02h receiving       \
      03h sending          > station being monitored
      04h initializing receive  /
  AH = keyboard lock status (00h unlocked, 01h locked)
Program: RATSR is a utility by James Berry provided with
    RemoteAccess/Professional, a commercial bulletin board system, that
```

```
    allows remote control of a station over a network
SeeAlso: INT 2D"AMIS"
--------d-2D--10-----------------------------
INT 2D - dLite 1.0+ - GET PARAMETER BLOCK ADDRESS
  AL = 10h
  AH = AMIS multiplex number for dLite
Return: CF clear if successful
      ES:BX -> parameter block (see #02577)
  CF set on error
Program: dLite is a shareware TSR by Rainer Schuetze which transparently
    expands compressed files when they are read
SeeAlso: AL=11h"dLite",AL=12h"dLite",INT 21/AX=FEDCh"PCMANAGE"

Format of dLite parameter block:
Offset  Size  Description (Table 02577)
 00h  BYTE  TSR flags (see #02578)
 01h  WORD  maximum number of programs needing original filesize
 03h  WORD  current number of programs needing original filesize
 05h  WORD  maximum number of files that can be handled by dLite (should
      be the same as FILES= in CONFIG.SYS)
 07h  WORD  offset (in the same segment as the parameter block) of the
      table of programs needing the original filesize (8 bytes
      each, without path or extension, uppercase, and zero \
      terminated if shorter than 8 bytes)

Bitfields for dLite TSR flags:
Bit(s)  Description (Table 02578)
 0  deny FCB access
 1  dLite sleeping rather than activated
 2  always indicate original filesize when reading directory entries,
    rather than only for specified programs
 3-7  reserved
SeeAlso: #02577
--------V-2D--10-----------------------------
INT 2D - Burnout Plus v3.00 - GET STATE/CONTROL INFORMATION
  AL = 10h
  AH = AMIS multiplex number for Burnout Plus
Return: AL = 01h
  BX = Burnout Plus status (see #02579)
  CX = record of features loaded (see #02580)
  ES:DI -> Burnout Plus control structure (see #02581)
```

```
Program: Burnout Plus is a DOS screen saver from Cove Software
SeeAlso: INT 14/AX=AA01h,INT 2D"AMIS"
Index:  screen saver;Burnout Plus

Bitfields for Burnout Plus status:
Bit(s)  Description (Table 02579)
 0  screen is blanked
 1  MS Windows is active (Burnout Plus deactivated)
 2-15 reserved

Bitfields for Burnout Plus features loaded/features enabled:
Bit(s)  Description (Table 02580)
 0  mouse activity monitor
 1  passkey support
 2  password support
 3  continuous clear
 4  software blanking
 5  video activity monitor
 6  disk activity monitor
 7  activating keystroke suppression
SeeAlso: #02581

Format of Burnout Plus control structure:
Offset  Size  Description (Table 02581)
 00h  BYTE  size of structure in bytes
 01h  WORD  Burnout Plus version
 03h  WORD  screen blanking reset count in clock ticks
 05h  WORD  current countdown value in clock ticks
 07h  BYTE  type of timeout specification
 08h  BYTE  instant-blank hotkey
 09h  WORD  extended status information (see #02582)
    the bits for password, passkey, and software blanking are
      ignored and cannot be enabled or disabled externally
 0Bh  WORD  features enabled (see #02580)
Note: all fields except the first two may be modified by external programs
    to affect the operation of Burnout Plus
Index:  hotkeys;Burnout Plus

Bitfields for extended Burnout Plus status information:
Bit(s)  Description (Table 02582)
 0  Burnout Plus disabled
```

```
 1  force screen to blank on next clock tick
 2  restore screen if currently blanked
 3-15 reserved
Note: 1 and 2 are automatically cleared by Burnout Plus after blanking
    or restoring the screen
SeeAlso: #02581
--------V-2D--10-----------------------------
INT 2D U - Screen Thief v1.00 - FREE HIGH MEMORY BUFFERS
  AL = 10h
  AH = AMIS multiplex number for Screen Thief
Return: nothing
Program: Screen Thief is a graphics screen grabber
Note: releases any code and data stored in EMS, DOS UMBs, or XMS UMBs, but
    does not release the low-memory stub; this may be used to effect a
    partial uninstall if INT 2D/AL=02h fails
SeeAlso: INT D8"Screen Thief"
--------i-2D--10-----------------------------
INT 2D U - RAMLIGHT v1.0 - GET MONITORING INFORMATION
  AL = 10h
  AH = AMIS multiplex number for RAMLIGHT
Return: ES:BX -> array of fake device driver headers used in monitoring
  CX = number of drives being monitored???
--------U-2D--10-----------------------------
INT 2D - DTown Utilities v1.40+ - EXTENDED API INSTALLATION CHECK
  AL = 10h
Return: AL = FFh
  BL = extended API availability (00h no, 01h API is loaded)
SeeAlso: INT 2D/AL=11h"DTown",INT 2D/AL=20h,INT 2D/AL=50h
--------s-2D--10-----------------------------
INT 2D - CDTSR - GET INTERNAL VARIABLE TABLE
  AL = 10h
  AH = AMIS multiplex number for CDTSR
Return: CX:DX -> CDTSR internal variable structure (see #02583)
Program: CDTSR is a resident audio CD player by Colin Hill
SeeAlso: INT 2D/AL=11h"CDTSR",INT 2D/AL=12h"CDTSR",INT 2D/AL=13h"CDTSR"

Format of CDTSR internal variable structure:
Offset  Size  Description (Table 02583)
 00h  BYTE  hotkey scan code (see #00006)
 01h  BYTE  hotkey shift states
 02h  BYTE  flag: repeat
```

```
 03h  BYTE  flag: custom repeat
 04h  BYTE  flag: background polling
 05h  DWORD (read-only) internal timing variable
 09h  DWORD current track play position, in frames
 0Dh  DWORD current disk play position, in frames
 11h  BYTE  number of entries in track program
 12h  BYTE  index into track program currently playing (FFh if not playing)
 13h 100 BYTEs  track program (each byte contains one track number)
 77h  BYTE  saved cursor end scan line
 78h  BYTE  saved cursor start scan line
 79h  BYTE  currently playing track
 7Ah  BYTE  CD driver media-change flag
 7Bh  WORD  video base segment during last popup
 7Dh  WORD  video page offset during last popup
 7Fh  BYTE  currently-selected track
 80h  DWORD begin of custom repeat, in frames
 84h  DWORD end of custom repeat, in frames
 88h  WORD  track program index of top list item
--------K-2D--10-----------------------------
INT 2D - KEYBIT Lite v5+ - GET POINTER TO STATUS BYTE
  AL = 10h
  AH = AMIS multiplex number for KEYBIT Lite
Return: DX:BX -> status byte (see #02584)
Program: KEYBIT Lite is an enhanced Italian keyboard driver by Pino Navato.
SeeAlso: INT 2D"AMIS"

Bitfields for KEYBIT Lite status byte:
Bit(s)  Description (Table 02584)
 7  KEYBIT Lite active
 6  E-mail support active
 5-0  reserved
Notes:  E-mail support is one of the original features of KEYBIT Lite.  It is
    the automatic conversion of the 8-bits ASCII chars produced by some
    keys available on Italian keyboards to couples of 7-bits chars.
  Message editors should always enable e-mail support, they should also
    restore its original status before exiting.
  The user can change both status bits by hotkeys.
--------K-2D--10-----------------------------
INT 2D - ALTMENU - GET POINTER TO KEY CODE
  AL = 10h
  AH = AMIS multiplex number for signature 'Pino Nav' 'ALTMENU '
```

```
Return: DX:BX -> WORD key code to insert in keyboard buffer on Alt-key tap
Program: Pino Navato's freeware ALTMENU permits activating the menu bar of
    any program by pressing the Alt key alone.
Notes:  The value in the key code word will be returned in AX by a call to
    INT 16/AH=00h after the Alt key is pressed by itself
  ALTMENU may be disabled by setting the key code equal to 0000h
SeeAlso: INT 16/AH=00h,INT 2D"AMIS"
--------d-2D--11-----------------------------
INT 2D - dLite 1.0+ - CHECK FOR dPressed FILE AND GET ORIGINAL SIZE
  AL = 11h
  AH = AMIS multiplex number for dLite
  BX = file handle
Return: CF clear if successful
      DX:AX = size of uncompressed file
  CF set on error (not dPressed file)
SeeAlso: AL=10h"dLite",AL=12h"dLite"
--------U-2D--11-----------------------------
INT 2D - DTown Utilities v1.40+ - UTILITY INSTALLATION CHECK
  AL = 11h
  BL = function
      00h get number of installed utilities
    Return: BL = number of utilities
      01h get installed utilities
    DX:DI -> buffer containing one byte for each utility
    Return: DX:DI buffer filled with flags (0=no,1=yes) indicating
        whether the corresponding utility is loaded
Return: AL = FFh if supported
Note: this function is only available if the extended API has been installed
    in the resident portion
SeeAlso: INT 2D/AL=10h"DTown",INT 2D/AL=20h
--------s-2D--11-----------------------------
INT 2D - CDTSR - REPROGRAM CDTSR
  AL = 11h
  AH = AMIS multiplex number for CDTSR
Return: nothing
Program: CDTSR is a resident audio CD player by Colin Hill
Desc: reprograms CDTSR based on the values in the internal variable
    structure (see #02583), which may have been changed by an application
SeeAlso: INT 2D/AL=10h"CDTSR",INT 2D/AL=12h"CDTSR",INT 2D/AL=13h"CDTSR"
--------d-2D--12-----------------------------
INT 2D - dLite 1.0+ - CHECK FOR dPressed FILE AND GET COMPRESSED SIZE
```

```
  AL = 12h
  AH = AMIS multiplex number for dLite
  BX = file handle
Return: CF clear if successful
      DX:AX = size of compressed file
  CF set on error (not dPressed file)
SeeAlso: AL=10h"dLite",AL=11h"dLite"
--------s-2D--12-----------------------------
INT 2D - CDTSR - DISABLE POPUP
  AL = 12h
  AH = AMIS multiplex number for CDTSR
Return: nothing
SeeAlso: INT 2D/AL=10h"CDTSR",INT 2D/AL=11h"CDTSR",INT 2D/AL=13h"CDTSR"
--------U-2D--12-----------------------------
INT 2D - FREEVER - GET ORIGINAL DOS VERSION INFO
  AL = 12h
  AH = AMIS multiplex number for FREEVER
Return: AL = FFh if successful
      BH = major DOS version
      BL = minor DOS version
      CH = DOS version flag
      CL = OEM number
      DH = major DR DOS version number (FFh if unknown)
      DL = minor DR DOS version number (FFh if unknown)
Program: FREEVER is an AMIS-conformant freeware DOS version-faking TSR similar
    to SETVER for any DOS-compatible OS, written by Matthias Paul
SeeAlso: INT 2D/AL=13h"FREEVER",INT 2D/AL=14h"FREEVER",INT 2D/AL=17h"FREEVER"
--------s-2D--13-----------------------------
INT 2D - CDTSR - ENABLE POPUP
  AL = 13h
  AH = AMIS multiplex number for CDTSR
Return: nothing
Program: CDTSR is a resident audio CD player by Colin Hill
SeeAlso: INT 2D/AL=10h"CDTSR",INT 2D/AL=11h"CDTSR",INT 2D/AL=12h"CDTSR"
--------U-2D--13-----------------------------
INT 2D - FREEVER - SET VERSION NUMBERS
  AL = 13h
  AH = AMIS multiplex number for FREEVER
  BH = new major DOS version
  BL = new minor DOS version
  CH = new DOS version flag
```

```
  CL = new DOS revision number
  DH = new OEM number
SeeAlso: INT 2D/AL=12h"FREEVER",INT 2D/AL=15h"FREEVER",INT 2D/AL=17h"FREEVER"
--------U-2D--14-----------------------------
INT 2D - FREEVER - ENABLE TSR
  AL = 14h
  AH = AMIS multiplex number for FREEVER
Return: AL = FFh if successful
SeeAlso: INT 2D/AL=12h"FREEVER",INT 2D/AL=15h"FREEVER",INT 2D/AL=16h"FREEVER"
--------U-2D--15-----------------------------
INT 2D - FREEVER - DISABLE TSR
  AL = 15h
  AH = AMIS multiplex number for FREEVER
Return: AL = FFh if successful
SeeAlso: INT 2D/AL=12h"FREEVER",INT 2D/AL=14h"FREEVER",INT 2D/AL=16h"FREEVER"
--------U-2D--16-----------------------------
INT 2D - FREEVER - GET TSR STATUS
  AL = 16h
  AH = AMIS multiplex number for FREEVER
Return: AL = FFh if successful
      BL = status
    01h resident and active
    02h resident and inactive
SeeAlso: INT 2D/AL=12h"FREEVER",INT 2D/AL=15h"FREEVER",INT 2D/AL=17h"FREEVER"
--------U-2D--17-----------------------------
INT 2D - FREEVER - GET TaskMAX STATUS AT INSTALLATION
  AL = 17h
  AH = AMIS multiplex number for FREEVER
Return: AL = FFh if successful
      BL = status
    00h if TaskMAX not loaded before SETDRVER
    FFh if TaskMAX was loaded before SETDRVER
Program: FREEVER is an AMIS-conformant freeware DOS version-faking TSR similar
    to SETVER for any DOS-compatible OS, written by Matthias Paul
SeeAlso: INT 2D/AL=12h"FREEVER",INT 2D/AL=14h"FREEVER",INT 2D/AL=16h"FREEVER"
--------U-2D--20-----------------------------
INT 2D - DTown Utilities v1.40+ - GET POP-UP HANDLER ADDRESS
  AL = 20h
Return: AL = FFh if available
      DX:DI -> DTU popup-handler
Program: DTown Utilities is a freeware programmer's utility TSR by Jeroen van
```

```
    Disseldorp
Note: this function is only available if the extended API has been installed
    in the resident portion
SeeAlso: INT 2D/AL=10h"DTown",INT 2D/AL=21h,INT 2D/AL=50h,INT 2D/AL=51h
SeeAlso: INT 03"DTown"
--------U-2D--21-----------------------------
INT 2D - DTown Utilities v1.40+ - POP UP
  AL = 21h
  BL = which utility to bring up
       00h active utility
       01h help screen
       0Ah ASCII table
       0Bh memory view
       0Ch CPU status
       0Dh calculator
       0Eh miscellaneous
       0Fh file viewer
       10h disassembler
Return: AL = status
       00h already active
       FFh popped up successfully
  BX = 0000h
Note: this function is only available if the extended API has been installed
    in the resident portion
SeeAlso: INT 2D/AL=10h"DTown",INT 2D/AL=20h,INT 2D/AL=50h
--------U-2D--50-----------------------------
INT 2D - DTown Utilities v1.40+ - MEMORY VIEW SET ADDRESS
  AL = 50h
  CX:DX = new address for start of memory view utility's display
Note: this function is only available if the extended API has been installed
    in the resident portion
Return: AL = FFh if supported
SeeAlso: INT 2D/AL=10h"DTown",INT 2D/AL=20h,INT 2D/AL=21h,INT 2D/AL=51h
--------U-2D--51-----------------------------
INT 2D - DTown Utilities v1.40+ - MEMORY VIEW SET REFERENCE
  AL = 51h
  BL = reference ("bookmark") number
  CX:DX = new address for reference
Return: AL = status
       00h invalid index
       FFh reference set
```

```
Note: this function is only available if the extended API has been installed
    in the resident portion
Program: DTown Utilities is a shareware programmer's utility TSR by Jeroen van
    Disseldorp
SeeAlso: INT 2D/AL=10h"DTown",INT 2D/AL=20h,INT 2D/AL=50h
--------b-2D--DC-----------------------------
INT 2D C - DAISY.SYS - BROADCAST: CHAIN RESCANNED
  AL = DCh
  AH = AMIS multiplex number for signature 'DAISYCHA' 'INDRIVER'
  DL = LPT Port Rescanned
Program: DAISY.SYS is a daisy chain manager for parallel port peripherals
    conforming to the IEEE 1284.3 Committee's daisy chain specification.
Desc: This Broadcast is sent whenever daisy chain IDs are reassigned to
    warn parallel port device drivers that their daisy chain ID may
    have been changed.
Note: This function is a callout from DAISY.SYS, NOT a call into DAISY.SYS
SeeAlso: INT 17/AX=0200h"Enhanced Parallel Port",#00633,#02570
--------l-2E---------------------------------
INT 2E U - DOS 2+ - PASS COMMAND TO COMMAND INTERPRETER FOR EXECUTION
  DS:SI -> commandline to execute (see #02585)
Return: all registers except CS:IP destroyed
  AX = status (4DOS v4.0)
     0000h successful
     FFFFh error before processing command (not enough memory, etc)
     other error number returned by command
Notes:  this call allows execution of arbitrary commands (including COMMAND.COM
    internal commands) without loading another copy of COMMAND.COM
  if COMMAND.COM is the user's command interpreter, the primary copy
    executes the command; this allows the master environment to be
    modified by issuing a "SET" command, but changes in the master
    environment will not become effective until all programs descended
    from the primary COMMAND.COM terminate
  since COMMAND.COM processes the string as if typed from the keyboard,
    the transient portion needs to be present, and the calling program
    must ensure that sufficient memory to load the transient portion can
    be allocated by DOS if necessary
  results are unpredictable if invoked by a program run from a batch file
    because this call is not reentrant and COMMAND.COM uses the same
    internal variables when processing a batch file
  hooked but ignored by 4DOS v3.0 COMMAND.COM replacement unless SHELL2E
    has been loaded
```

```
  the MS-DOS 5 Programmer's Reference calls this "Reload Transient"

Format of DOS commandline:
Offset  Size  Description (Table 02585)
 00h  BYTE  length of command string, not counting trailing CR
 01h  var command string
  N BYTE  0Dh (CR)
--------O-2E---------------------------------
INT 2E UP - Windows NT - NATIVE API
  EAX = function number (see #02586)
  EDX = address of parameter block
Return: ???

(Table 02586)
Values for Windows NT NTOS function number:
 000h AcceptConnectPort (24 bytes of parameters)
 001h AccessCheck   (32 bytes of parameters)
 002h AccessCheckAndAuditAlarm  (44 bytes of parameters)
 003h AddAtom     (8 bytes of parameters)
 004h AdjustGroupsToken (24 bytes of parameters)
 005h AdjustPrivilegesToken (24 bytes of parameters)
 006h AlertResumeThread (8 bytes of parameters)
 007h AlertThread   (4 bytes of parameters)
 008h AllocateLocallyUniqueId (4 bytes of parameters)
 009h AllocateUuids   (12 bytes of parameters)
 00Ah AllocateVirtualMemory (24 bytes of parameters)
 00Bh CallbackReturn    (12 bytes of parameters)
 00Ch CancelIoFile    (8 bytes of parameters)
 00Dh CancelTimer   (8 bytes of parameters)
 00Eh ClearEvent    (4 bytes of parameters)
 00Fh Close     (4 bytes of parameters)
 010h CloseObjectAuditAlarm (12 bytes of parameters)
 011h CompleteConnectPort (4 bytes of parameters)
 012h ConnectPort   (32 bytes of parameters)
 013h Continue    (8 bytes of parameters)
 014h CreateDirectoryObject (12 bytes of parameters)
 015h CreateEvent   (20 bytes of parameters)
 016h CreateEventPair   (12 bytes of parameters)
 017h CreateFile    (44 bytes of parameters)
 018h CreateIoCompletion  (16 bytes of parameters)
 019h CreateKey   (28 bytes of parameters)
```

```
01Ah CreateMailslotFile  (32 bytes of parameters)
01Bh CreateMutant    (16 bytes of parameters)
01Ch CreateNamedPipeFile (56 bytes of parameters)
01Dh CreatePagingFile  (16 bytes of parameters)
01Eh CreatePort    (20 bytes of parameters)
01Fh CreateProcess   (32 bytes of parameters)
020h CreateProfile   (36 bytes of parameters)
021h CreateSection   (28 bytes of parameters)
022h CreateSemaphore   (20 bytes of parameters)
023h CreateSymbolicLinkObject  (16 bytes of parameters)
024h CreateThread    (32 bytes of parameters)
025h CreateTimer   (16 bytes of parameters)
026h CreateToken   (52 bytes of parameters)
027h DelayExecution    (8 bytes of parameters)
028h DeleteAtom    (4 bytes of parameters)
029h DeleteFile    (4 bytes of parameters)
02Ah DeleteKey   (4 bytes of parameters)
02Bh DeleteObjectAuditAlarm  (12 bytes of parameters)
02Ch DeleteValueKey    (8 bytes of parameters)
02Dh DeviceIoControlFile (40 bytes of parameters)
02Eh DisplayString   (4 bytes of parameters)
02Fh DuplicateObject   (28 bytes of parameters)
030h DuplicateToken    (24 bytes of parameters)
031h EnumerateKey    (24 bytes of parameters)
032h EnumerateValueKey (24 bytes of parameters)
033h ExtendSection   (8 bytes of parameters)
034h FindAtom    (8 bytes of parameters)
035h FlushBuffersFile  (8 bytes of parameters)
036h FlushInstructionCache (12 bytes of parameters)
037h FlushKey    (4 bytes of parameters)
038h FlushVirtualMemory  (16 bytes of parameters)
039h FlushWriteBuffer  (no parameters)
03Ah FreeVirtualMemory (16 bytes of parameters)
03Bh FsControlFile   (40 bytes of parameters)
03Ch GetContextThread  (8 bytes of parameters)
03Dh GetPlugPlayEvent  (16 bytes of parameters)
03Eh GetTickCount    (no parameters)
03Fh ImpersonateClientOfPort (8 bytes of parameters)
040h ImpersonateThread (12 bytes of parameters)
041h InitializeRegistry  (4 bytes of parameters)
042h ListenPort    (8 bytes of parameters)
```

```
043h LoadDriver    (4 bytes of parameters)
044h LoadKey (8 bytes of parameters)
045h LoadKey2    (12 bytes of parameters)
046h LockFile    (40 bytes of parameters)
047h LockVirtualMemory (16 bytes of parameters)
048h MakeTemporaryObject (4 bytes of parameters)
049h MapViewOfSection  (40 bytes of parameters)
04Ah NotifyChangeDirectoryFile (36 bytes of parameters)
04Bh NotifyChangeKey   (40 bytes of parameters)
04Ch OpenDirectoryObject (12 bytes of parameters)
04Dh OpenEvent   (12 bytes of parameters)
04Eh OpenEventPair   (12 bytes of parameters)
04Fh OpenFile    (24 bytes of parameters)
050h OpenIoCompletion  (12 bytes of parameters)
051h OpenKey     (12 bytes of parameters)
052h OpenMutant    (12 bytes of parameters)
053h OpenObjectAuditAlarm  (48 bytes of parameters)
054h OpenProcess   (16 bytes of parameters)
055h OpenProcessToken  (12 bytes of parameters)
056h OpenSection   (12 bytes of parameters)
057h OpenSemaphore   (12 bytes of parameters)
058h OpenSymbolicLinkObject  (12 bytes of parameters)
059h OpenThread    (16 bytes of parameters)
05Ah OpenThreadToken   (16 bytes of parameters)
05Bh OpenTimer   (12 bytes of parameters)
05Ch PlugPlayControl   (16 bytes of parameters)
05Dh PrivilegeCheck    (12 bytes of parameters)
05Eh PrivilegedServiceAuditAlarm (20 bytes of parameters)
05Fh PrivilegeObjectAuditAlarm (24 bytes of parameters)
060h ProtectVirtualMemory  (20 bytes of parameters)
061h PulseEvent    (8 bytes of parameters)
062h QueryInformationAtom  (20 bytes of parameters)
063h QueryAttributesFile (8 bytes of parameters)
064h QueryDefaultLocale  (8 bytes of parameters)
065h QueryDirectoryFile  (44 bytes of parameters)
066h QueryDirectoryObject  (28 bytes of parameters)
067h QueryEaFile   (36 bytes of parameters)
068h QueryEvent    (20 bytes of parameters)
069h QueryFullAttributesFile (8 bytes of parameters)
06Ah QueryInformationFile  (20 bytes of parameters)
06Bh QueryIoCompletion (20 bytes of parameters)
```

```
06Ch QueryInformationPort  (20 bytes of parameters)
06Dh QueryInformationProcess (20 bytes of parameters)
06Eh QueryInformationThread  (20 bytes of parameters)
06Fh QueryInformationToken (20 bytes of parameters)
070h QueryIntervalProfile  (8 bytes of parameters)
071h QueryKey    (20 bytes of parameters)
072h QueryMultipleValueKey (24 bytes of parameters)
073h QueryMutant   (20 bytes of parameters)
074h QueryObject   (20 bytes of parameters)
075h QueryOleDirectoryFile (44 bytes of parameters)
076h QueryPerformanceCounter (8 bytes of parameters)
077h QuerySection    (20 bytes of parameters)
078h QuerySecurityObject (20 bytes of parameters)
079h QuerySemaphore    (20 bytes of parameters)
07Ah QuerySymbolicLinkObject (12 bytes of parameters)
07Bh QuerySystemEnvironmentValue (16 bytes of parameters)
07Ch QuerySystemInformation  (16 bytes of parameters)
07Dh QuerySystemTime   (4 bytes of parameters)
07Eh QueryTimer    (20 bytes of parameters)
07Fh QueryTimerResolution  (12 bytes of parameters)
080h QueryValueKey   (24 bytes of parameters)
081h QueryVirtualMemory  (24 bytes of parameters)
082h QueryVolumeInformationFile  (20 bytes of parameters)
083h QueueApcThread    (20 bytes of parameters)
084h RaiseException    (12 bytes of parameters)
085h RaiseHardError    (24 bytes of parameters)
086h ReadFile    (36 bytes of parameters)
087h ReadFileScatter   (36 bytes of parameters)
088h ReadRequestData   (24 bytes of parameters)
089h ReadVirtualMemory (20 bytes of parameters)
08Ah RegisterThreadTerminatePort (4 bytes of parameters)
08Bh ReleaseMutant   (8 bytes of parameters)
08Ch ReleaseSemaphore  (12 bytes of parameters)
08Dh RemoveIoCompletion  (20 bytes of parameters)
08Eh ReplaceKey    (12 bytes of parameters)
08Fh ReplyPort   (8 bytes of parameters)
090h ReplyWaitReceivePort  (16 bytes of parameters)
091h ReplyWaitReplyPort  (8 bytes of parameters)
092h RequestPort   (8 bytes of parameters)
093h RequestWaitReplyPort  (12 bytes of parameters)
094h ResetEvent    (8 bytes of parameters)
```

```
095h RestoreKey    (12 bytes of parameters)
096h ResumeThread    (8 bytes of parameters)
097h SaveKey (8 bytes of parameters)
098h SetIoCompletion   (20 bytes of parameters)
099h SetContextThread  (8 bytes of parameters)
09Ah SetDefaultHardErrorPort (4 bytes of parameters)
09Bh SetDefaultLocale  (8 bytes of parameters)
09Ch SetEaFile   (16 bytes of parameters)
09Dh SetEvent    (8 bytes of parameters)
09Eh SetHighEventPair  (4 bytes of parameters)
09Fh SetHighWaitLowEventPair (4 bytes of parameters)
0A0h ???     (??? bytes of parameters)
0A1h SetInformationFile  (20 bytes of parameters)
0A2h SetInformationKey (16 bytes of parameters)
0A3h SetInformationObject  (16 bytes of parameters)
0A4h SetInformationProcess (16 bytes of parameters)
0A5h SetInformationThread  (16 bytes of parameters)
0A6h SetInformationToken (16 bytes of parameters)
0A7h SetIntervalProfile  (8 bytes of parameters)
0A8h SetLdtEntries   (24 bytes of parameters)
0A9h SetLowEventPair   (4 bytes of parameters)
0AAh SetLowWaitHighEventPair (4 bytes of parameters)
0ABh ???     (??? bytes of parameters)
0ACh SetSecurityObject (12 bytes of parameters)
0ADh SetSystemEnvironmentValue (8 bytes of parameters)
0AEh SetSystemInformation  (12 bytes of parameters)
0AFh SetSystemPowerState (12 bytes of parameters)
0B0h SetSystemTime   (8 bytes of parameters)
0B1h SetTimer    (28 bytes of parameters)
0B2h SetTimerResolution  (12 bytes of parameters)
0B3h SetValueKey   (24 bytes of parameters)
0B4h SetVolumeInformationFile  (20 bytes of parameters)
0B5h ShutdownSystem    (4 bytes of parameters)
0B6h SignalAndWaitForSingleObject  (16 bytes of parameters)
0B7h StartProfile    (4 bytes of parameters)
0B8h StopProfile   (4 bytes of parameters)
0B9h SuspendThread   (8 bytes of parameters)
0BAh SystemDebugControl  (24 bytes of parameters)
0BBh TerminateProcess  (8 bytes of parameters)
0BCh TerminateThread   (8 bytes of parameters)
0BDh TestAlert   (no parameters)
```

```
 0BEh UnloadDriver    (4 bytes of parameters)
 0BFh UnloadKey   (4 bytes of parameters)
 0C0h UnlockFile    (20 bytes of parameters)
 0C1h UnlockVirtualMemory (16 bytes of parameters)
 0C2h UnmapViewOfSection  (8 bytes of parameters)
 0C3h VdmControl    (8 bytes of parameters)
 0C4h WaitForMultipleObjects  (20 bytes of parameters)
 0C5h WaitForSingleObject (12 bytes of parameters)
 0C6h WaitHighEventPair (4 bytes of parameters)
 0C7h WaitLowEventPair  (4 bytes of parameters)
 0C8h WriteFile   (36 bytes of parameters)
 0C9h WriteFileGather   (36 bytes of parameters)
 0CAh WriteRequestData  (24 bytes of parameters)
 0CBh WriteVirtualMemory  (20 bytes of parameters)
 0CCh W32Call     (20 bytes of parameters)
 0CDh CreateChannel   (8 bytes of parameters)
 0CEh ListenChannel   (8 bytes of parameters)
 0CFh OpenChannel   (8 bytes of parameters)
 0D0h ReplyWaitSendChannel  (12 bytes of parameters)
 0D1h SendWaitReplyChannel  (16 bytes of parameters)
 0D2h SetContextChannel (4 bytes of parameters)
 0D3h YieldExecution    (no parameters)
--------l-2E----BXE22E-----------------------
INT 2E - 4DOS v2.x-3.03 SHELL2E.COM - UNINSTALL
  BX = E22Eh
  DS:SI -> zero byte
Return: if successful, SHELL2E terminates itself with INT 21/AH=4Ch
----------2F---------------------------------
INT 2F - Multiplex - NOTES
  AH = identifier of program which is to handle the interrupt
     00h-3Fh reserved for IBM (for DOS)
     40h-7Fh reserved for Microsoft (for DOS)
     80h-B7h reserved for IBM
     B8h-BFh reserved for networks
     C0h-FFh reserved for applications
  AL is the function code
   This is a general mechanism for verifying the presence of a TSR and
   communicating with it.  When searching for a free identifier code for AH
   using the installation check (AL=00h), the calling program should set
   BX/CX/DX to 0000h and must not depend on any registers other than CS:IP
   and SS:SP to be valid on return, since numerous programs now use additional
```

```
   registers on input and/or output for the installation check.
Notes:  Since the multiplex chain is growing so long, and beginning to
    experience multiplex number collisions, I have proposed an alternate
    multiplex interrupt on INT 2D.  If you decide to use the alternate
    multiplex, please let me know.
  DOS and some other programs return values in the flags register, so
    any TSR which chains by calling the previous handler rather than
    jumping to it should ensure that the returned flags are preserved
    and passed back to the original caller
SeeAlso: INT 2D"ALTERNATE MULTIPLEX"
--------t-2F---------------------------------
INT 2F - BMB Compuscience Canada Utilities Interface - INSTALLATION CHECK
  AH = xx (dynamically assigned based upon a search for a multiplex
     number which doesn't answer installed)
  AL = 00h installation check
  ES:DI = EBEBh:BEBEh
Return: AL = 00h not installed
       01h not installed, not OK to install
       FFh installed; if ES:DI was EBEBh:BEBEh on entry, ES:DI will point
     to a string of the form 'MMMMPPPPPPPPvNNNN' where MMMM is a
     short form of the manufacturer's name, PPPPPPPP is a product
     name and NNNN is the product's version number
--------t-2F---------------------------------
INT 2F - Ross Wentworth's Turbo Pascal POPUP LIBRARY
  AH = programmer-selected multiplex number
  AL = function
      00h installation check
    Return: AL = FFh if installed
      01h get TSR interrupt vectors
    Return: DX:AX -> vector table (see #02587)
      02h get TSR code segment
    Return: AX = code segment for all interrupt handlers
      03h call user exit routine and release TSR's memory
      04h get signature string
    Return: DX:AX -> counted string containing signature
      05h get TSR's INT 2F handler
    Return: DX:AX -> INT 2F handler
      06h enable/disable TSR
    BL = new state (00h disabled, 01h enabled)
      07h activate TSR (popup if not disabled)
      08h get hotkeys
```

```
    BL = which hotkey (00h = hotkey 1, 01h = hotkey 2)
    Return: AX = hotkey (AH = keyflags, AL = scancode)
      09h set hotkey
    BL = which hotkey (00h = hotkey 1, 01h = hotkey 2)
    CX = new hotkey (CH = keyflags, CL = scancode)
      0Ah-1Fh reserved
Index:  installation check;Ross Wentworth POPUP library
Index:  hotkeys;Ross Wentworth POPUP library

Format of POPUP vector table entry:
Offset  Size  Description (Table 02587)
 00h  BYTE  vector number (00h = end of table)
 01h  DWORD original vector
 05h  WORD  offset of interrupt handler in TSR's code segment
--------t-2F---------------------------------
INT 2F - CiriSOFT Spanish University of Valladolid TSR's Interface
  AH = xx (dynamically assigned based upon a search for a multiplex
     number from C0h to FFh which doesn't answer installed)
  AL = 00h installation check
  ES:DI = 1492h:1992h
Return: AL = 00h not installed
       01h not installed, not OK to install
       FFh installed; and if ES:DI was 1492h:1992h on entry, ES:DI will
       point to author_name_ver table (see #02588)
  AH = FFh
Note: this interface permits advanced communication with TSRs: it is possible
    to make a generic uninstall utility, advanced TSR relocator programs
    in order to fit fragmented memory areas, etc.
See also: INT 2D"AMIS",INT 2F"Compuscience"
Index:  installation check;CiriSOFT TSR interface
Index:  uninstall;CiriSOFT TSR interface

Format of CiriSOFT author_name_ver table:
Offset  Size  Description (Table 02588)
 -16  WORD  segment of the start of the resident TSR code (CS in programs
      with PSP, XMS upper memory segment if installed as UMB...)
 -14  WORD  offset of the start of the resident TSR code (frequently 100h
      in *.COM programs and 0 in upper memory TSR's).
 -12  WORD  memory used by TSR (in paragraphs). Knowing the memory area
      used by TSR is possible to determine if hooked vectors are
      still pointing it (and if it is safe to uninstall).
```

```
 -10  BYTE  characteristics byte (see #02589)
 -9 BYTE  number of multiplex entry used (redefinition available). Note
      that the TSR must use THIS variable in it's INT 2Fh handler.
 -8 WORD  offset to vector_area table (see #02590)
 -6 WORD  offset to extra_area table (see #02591,#02589 [bit 7])
 -4   4 BYTEs signature string "*###*"
 00h  var "AUTHOR:PROGRAM_NAME:VERSION",0  (variable length, this area
      is used in order to determine if the TSR is already resident
      and it's version code; the ':' char is used as delimiter)

Bitfields for CiriSOFT characteristics byte:
Bit(s)  Description (Table 02589)
 0-2  type
  000 normal program (with PSP)
  001 upper XMS memory block (needed HIMEM.SYS function to free memory
        when uninstalling)
  010 device driver (*.SYS)
  011 device driver in EXE format
  1xx others (reserved)
 3-6  reserved
 7  set if extra_table defined and supported (see #02591)
SeeAlso: #02588

Format of CiriSOFT vector_area table:
Offset  Size  Description (Table 02590)
 -1 BYTE  number of vectors intercepted by TSR
 00h  BYTE  first vector number
 01h  DWORD first vector pointer before installing the TSR
 05h  BYTE  second vector number
 06h  DWORD second vector pointer before installing the TSR
 0Ah  ... (and so on)
Note: the TSR must use these variables to invoke the previous interrupt
    handler routines
SeeAlso: #02588

Format of extra_area table (needed only to improve relocation feature):
Offset  Size  Description (Table 02591)
 00h  WORD  offset to external_ctrl table (see #02592)
    0000h if not supported
 02h  WORD  reserved for future use (0)
SeeAlso: #02588
```

```
Format of CiriSOFT external_ctrl table:
Offset  Size  Description (Table 02592)
 00h  BYTE  bit 0: TSR is relocatable (no absolute segment references)
 01h  WORD  offset to a variable which can activate/inhibit the TSR
 ---And if bit 0 in offset 00h is off:
 03h  DWORD pointer to ASCIZ pathname for executable file which supports
      /SR parameter (silent installation & inhibit)
 07h  DWORD pointer to first variable to initialize on the copy reloaded
      from the previous TSR still resident
 0Bh  DWORD pointer to last variable (all variables packed in one block)
--------c-2F00--------------------------------
INT 2F U - DOS 2.x only PRINT.COM - SUBMIT FILE FOR PRINTING
  AH = 00h
  DS:DX -> opened FCB (see #01345 at INT 21/AX=0Fh)
Return: AH = number of files currently in print queue
  AL = status
       00h file successfully added
       01h queue is full
  ES:BX -> print queue (10 FCBs; first byte of FFh indicates unused)
  ES:DX -> currently-printing FCB (if DX=FFFFh, none printing)
Notes:  DOS 2.x PRINT.COM does not chain to previous INT 2F handler
  values in AH other than 00h or 01h cause PRINT to return the number of
    files in the queue in AH
SeeAlso: AH=01h"PRINT",AX=0102h
--------P-2F00--------------------------------
INT 2F U - PSPRINT - PRINT JOB CONTROL
  AH = 00h
  ???
Return: ???
--------c-2F0080------------------------------
INT 2F - DOS 3.1+ PRINT - GIVE PRINT A TIME SLICE
  AX = 0080h
Return: after PRINT executes
Notes:  PRINT returns AL=01h if AH=00h but AL is not 80h on entry
  this function is not supported by the Novell DOS 7 PRINT.COM
--------N-2F00D8------------------------------
INT 2F - Personal NetWare - VLM - ???
  AX = 00D8h
  ???
Return: ???
```

```
Note: hooked by one of the .VLMs loaded by VLM.EXE v1.10, but apparently a
    NOP
--------c-2F01-------------------------------
INT 2F U - DOS 2.x only PRINT.COM - REMOVE FILE FROM PRINT QUEUE
  AH = 01h
  DS:DX -> FCB (see #01345 at INT 21/AH=0Fh) for file to be canceled
Return: AH = number of files currently in print queue
  AL = 00h (successful)
  ES:BX -> print queue (10 FCBs; first byte of FFh indicates unused)
  ES:DX -> currently-printing FCB (if DX=FFFFh, none printing)
Notes:  DOS 2.x PRINT.COM does not chain to previous INT 2F handler
  values in AH other than 00h or 01h cause PRINT to return the number of
    files in the queue in AH
SeeAlso: AH=00h"PRINT.COM",AX=0103h
--------c-2F0100-----------------------------
INT 2F - DOS 3.0+ PRINT - INSTALLATION CHECK
  AX = 0100h
Return: AL = status
      00h not installed
      01h not installed, but not OK to install
      FFh installed
    AH = 00h (Novell DOS 7)
SeeAlso: AX=0101h
--------c-2F0100SI20D6-----------------------
INT 2F U - PrintCache 3.1 PRINT.COM - INSTALLATION CHECK
  AX = 0100h
  SI = 20D6h
  DI = 8761h
Return: AX = 00FFh if installed
  DI = 0001h if PrintCache's PRINT.COM installed and magic values match
      SI = resident code segment
Program: PrintCache PRINT.COM is a DOS PRINT replacement included in
    LaserTools' PrintCache memory/disk-based print spooler package
Note: if either of SI or DI differ from the indicated magic values, only AX
    will be modified on return, for compatibility with DOS PRINT
SeeAlso: AX=0101h/SI=20D6h,AX=C000h"PCACHE"
--------c-2F0101-----------------------------
INT 2F - DOS 3.0+ PRINT - SUBMIT FILE FOR PRINTING
  AX = 0101h
  DS:DX -> submit packet (see #02593)
Return: CF clear if successful
```

```
      AL = status
    01h added to queue
    9Eh now printing
  CF set on error
      AX = error code (see #02594,#01680 at INT 21/AH=59h/BX=0000h)
SeeAlso: AX=0102h

Format of PRINT submit packet:
Offset  Size  Description (Table 02593)
 00h  BYTE  level (must be 00h)
 01h  DWORD pointer to ASCIZ filename (no wildcards)

(Table 02594)
Values for PRINT error code:
 0001h  invalid function
 0002h  file not found
 0003h  path not found
 0004h  out of file handles
 0005h  access denied
 0008h  print queue full
 0009h  spooler busy
 000Ch  name too long
 000Fh  invalid drive
--------c-2F0101SI20D6-----------------------
INT 2F U - PrintCache v3.1 PRINT.COM - SUBMIT FILE FOR PRINTING
  AX = 0101h
  SI = 20D6h
  DI = 8761h
  DS:DX -> submit packet (see #02593)
  CL = print options
      bit 4: use default options
Return: CF clear if successful
      AL = status
    01h added to queue
    9Eh now printing
  CF set on error
      AX = error code (see #02594)
Program: PrintCache PRINT.COM is a DOS PRINT replacement included in
    LaserTools' PrintCache memory/disk-based print spooler package
Note: if either SI or DI differs from the indicated magic values on entry,
    PrintCache will use the default print options for the file for
```

```
    compatibility with DOS PRINT
SeeAlso: AX=0100h/SI=20D6h,AX=0101h,AH=00h"PRINT",AX=0107h"PrintCache"
--------c-2F0102-----------------------------
INT 2F - DOS 3.0+ PRINT - REMOVE FILE FROM PRINT QUEUE
  AX = 0102h
  DS:DX -> ASCIZ filename (wildcards allowed)
Return: CF clear if successful
  CF set on error
      AX = error code (see #02594)
SeeAlso: AX=0101h,AX=0103h,AH=01h"PRINT"
--------c-2F0103-----------------------------
INT 2F - DOS 3.0+ PRINT - CANCEL ALL FILES IN PRINT QUEUE
  AX = 0103h
Return: CF clear if successful
  CF set on error
      AX = error code (see #02594)
SeeAlso: AX=0102h
--------c-2F0104-----------------------------
INT 2F - DOS 3.0+ PRINT - FREEZE PRINT QUEUE TO READ JOB STATUS
  AX = 0104h
Return: CF clear if successful
      DX = error count since status last read
      DS:SI -> print queue
  CF set on error
      AX = error code (see #02594)
Desc: get the list of print jobs, temporarily suspending PRINT's activities
    to avoid changing the list while it is being examined
Notes:  the print queue is an array of 64-byte ASCIZ filenames terminated by
    an empty filename; the first name is the file currently being printed
  printing is stopped until AX=0105h is called to prevent the queue
    from changing while the filenames are being read
SeeAlso: AX=0101h,AX=0105h
--------c-2F0105-----------------------------
INT 2F - DOS 3.0+ PRINT - RESTART PRINT QUEUE AFTER STATUS READ
  AX = 0105h
Return: CF clear if successful
  CF set on error
      AX = error code (see #02594)
Desc: restart PRINT's activities once an application finishes examining the
    print queue
SeeAlso: AX=0104h
```

```
--------c-2F0106-----------------------------
INT 2F - DOS 3.3+ PRINT - GET PRINTER DEVICE
  AX = 0106h
Return: CF set if files in print queue
      AX = error code 0008h (queue full)
      DS:SI -> device driver header (see #01646)
  CF clear if print queue empty
      AX = 0000h
Desc: determine which device, if any, PRINT is currently using for output
Notes:  undocumented prior to the release of MS-DOS 5.0
  this function can be used to allow a program to avoid printing to the
    printer on which PRINT is currently performing output
SeeAlso: AX=0104h
--------c-2F0107-----------------------------
INT 2F U - PrintCache v3.1 PRINT.COM - SET TRAILING FORM FEEDS
  AX = 0107h
  CL bit 0: output form feed between print jobs
Return: AL destroyed
SeeAlso: AX=0100h/SI=20D6h,AX=0101h/SI=20D6h
--------N-2F0200-----------------------------
INT 2F U - PC LAN PROGRAM REDIR/REDIRIFS internal - INSTALLATION CHECK
  AX = 0200h
Return: AL = FFh if installed
Desc: determine whether the PC LAN Program redirector is installed
SeeAlso: AX=0201h,AX=0203h
--------N-2F0201-----------------------------
INT 2F U - PC LAN PROGRAM REDIR/REDIRIFS internal - ???
  AX = 0201h
Return: nothing???
Notes:  this function is called by the DOS 3.3+ PRINT.COM
  AX=0202h appears to be the opposite function
  these functions are supposedly used to signal opening and closing of
    printers
SeeAlso: AX=0202h
--------N-2F0202-----------------------------
INT 2F U - PC LAN PROGRAM REDIR/REDIRIFS internal - ???
  AX = 0202h
  ???
Return: nothing???
Notes:  this function is called by the DOS 3.3+ PRINT.COM
  these functions are supposedly used to signal opening and closing of
```

```
    printers
SeeAlso: AX=0201h
--------N-2F0203-----------------------------
INT 2F U - PC LAN PROGRAM REDIR/REDIRIFS internal - ???
  AX = 0203h
Return: nothing???
Notes:  this function is called by the DOS 3.3+ PRINT.COM
  AX=0204h appears to be the opposite function
  these functions are supposedly used to signal opening and closing of
    printers
SeeAlso: AX=0200h,AX=0204h
--------N-2F0204-----------------------------
INT 2F U - PC LAN PROGRAM REDIR/REDIRIFS internal - ???
  AX = 0204h
  ???
Return: nothing???
Notes:  this function is called by the DOS 3.3+ PRINT.COM
  AX=0203h appears to be the opposite function
  these functions are supposedly used to signal opening and closing of
    printers
SeeAlso: AX=0200h,AX=0203h
--------N-2F---------------------------------
INT 2F U - PC LAN PROGRAM REDIR/REDIRIFS internal - ???
  AX = 02xxh
  ???
Return: ???
--------l-2F0500-----------------------------
INT 2F U - DOS 3.0+ CRITICAL ERROR HANDLER - INSTALLATION CHECK
  AX = 0500h
Return: AL = status
      00h not installed, OK to install
      01h not installed, can't install
      FFh installed
  BX destroyed (MSCDEX v2.21-2.25)
  CF clear (MSCDEX v2.21-2.25)
Desc: determine whether a critical error message override is installed
Note: this set of functions allows a user program to partially or completely
    override the default critical error handler's message in COMMAND.COM
SeeAlso: AH=05h,INT 24
--------l-2F05-----------------------------
INT 2F CU - DOS 3.0+ CRITICAL ERROR HANDLER - EXPAND ERROR INTO STRING
```

```
  AH = 05h
---DOS 3.x---
  AL = extended error code (not zero)
---DOS 4.0+ ---
  AL = error type
       01h DOS extended error code
       02h parameter error
  BX = error code
Return: CF clear if successful
       ES:DI -> ASCIZ error message (read-only)
       AL = completion state
     00h message requires completion with device name, drive, etc.
     01h message is complete as returned
  CF set if error code can't be converted to string
       AX,DI,ES destroyed
  other flags corrupted
Notes:  called at start of COMMAND.COM's default critical error handler if
     installed by a user program, allowing partial or complete overriding
     of the default error messages
  subfunction 02h is called by many DOS 4 external programs
  DR DOS's COMMAND.COM appends additional info ("0 files copied") to the
     returned string
SeeAlso: AX=0500h,AX=122Eh,INT 24
--------U-2F0600-----------------------------
INT 2F - DOS 3.0+ ASSIGN - INSTALLATION CHECK
  AX = 0600h
Return: AL = status
       00h not installed
       01h not installed, but not OK to install
       FFh installed
Notes:  ASSIGN is not a TSR in DR DOS 5.0; it is internally replaced by SUBST
     (see INT 21/AH=52h)
  undocumented prior to the release of DOS 5.0
SeeAlso: AX=0601h,INT 21/AH=52h
--------U-2F0601-----------------------------
INT 2F U - DOS 3.0+ ASSIGN - GET DRIVE ASSIGNMENT TABLE
  AX = 0601h
Return: ES = segment of ASSIGN work area and assignment table
Note: the 26 bytes starting at ES:0103h specify which drive each of A: to Z:
     is mapped to.  Initially set to 01h 02h 03h....
SeeAlso: AX=0600h,AX=AF14h"WinDOS"
```

```
--------D-2F0800-----------------------------
INT 2F U - DRIVER.SYS support - INSTALLATION CHECK
  AX = 0800h
Return: AL = status
      00h not installed, OK to install
      01h not installed, not OK to install
      FFh installed
Desc: determine whether the internal support code used by DRIVER.SYS is
    present; it is always present in DOS 3.2+
Note: supported by DR DOS 5.0 and Novell DOS 7
--------D-2F0801-----------------------------
INT 2F U - DRIVER.SYS support - ADD NEW BLOCK DEVICE
  AX = 0801h
  DS:DI -> drive data table (see #02601,#02602,#02603)
Return: AX,BX,SI,ES destroyed
Notes:  moves down internal list of drive data tables, copying and modifying
    the drive description flags word for tables referencing same physical
    drive
  the data table is appended to the chain of tables
  supported by DR DOS 5.0 and Novell DOS 7
SeeAlso: AX=0803h
--------D-2F0802-----------------------------
INT 2F U - DRIVER.SYS support - EXECUTE DEVICE DRIVER REQUEST
  AX = 0802h
  ES:BX -> device driver request header (see #02597)
Return: request header updated as per requested operation
  STACK:  WORD  original flags from INT call (left by RETF in device
        driver, at least in DOS 5.0-6.22)
Notes:  supported by DR DOS 5.0
  DOS 3.2 executes this function on any AL value from 02h through F7h;
    DOS 4.0+ executes this function on AL=02h and AL=04h-F7h
  the command codes (see #02595) and structures described below apply
    to all drivers which support the appropriate commands; this call is
    just one of a number of ways in which a device driver request may
    be invoked
  supported by Novell DOS 7
SeeAlso: AX=0800h,AX=0801h,AX=0803h,AX=1510h,INT 21/AH=52h,INT 21/AH=99h
SeeAlso: INT 21/AH=9Ah

(Table 02595)
Values for device driver command code:
```

```
 00h (0)  INIT
 01h (1)  MEDIA CHECK (block devices)
 02h (2)  BUILD BPB (block devices)
 03h (3)  IOCTL INPUT
 04h (4)  INPUT
 05h (5)  NONDESTRUCTIVE INPUT, NO WAIT (character devices)
 06h (6)  INPUT STATUS (character devices)
 07h (7)  INPUT FLUSH (character devices)
 08h (8)  OUTPUT
 09h (9)  OUTPUT WITH VERIFY
 0Ah (10) OUTPUT STATUS (character devices)
 0Bh (11) OUTPUT FLUSH (character devices)
 0Ch (12) IOCTL OUTPUT
 0Dh (13) (DOS 3.0+) DEVICE OPEN
 0Eh (14) (DOS 3.0+) DEVICE CLOSE
 0Fh (15) (DOS 3.0+) REMOVABLE MEDIA (block devices)
 10h (16) (DOS 3.0+) OUTPUT UNTIL BUSY (character devices)
 11h (17) (European MS-DOS 4.0) STOP OUTPUT (console screen drivers only)
 12h (18) (European MS-DOS 4.0) RESTART OUTPUT (console screen drivers only)
 13h (19) (DOS 3.2+) GENERIC IOCTL
 14h (20) (DOS 4.0, KKCFUNC) DEVICE RESTORE (character device)
 15h (21) (European MS-DOS 4.0) RESET UNCERTAIN MEDIA FLAG
 16h (22) (DOS 4.0) unknown???
 17h (23) (DOS 3.2+) GET LOGICAL DEVICE
 18h (24) (DOS 3.2+) SET LOGICAL DEVICE
 19h (25) (DOS 5.0+) CHECK GENERIC IOCTL SUPPORT
 80h (128)(CD-ROM,DRFAT32) READ LONG
 81h (129)(CD-ROM) reserved
 82h (130)(CD-ROM,DRFAT32) READ LONG PREFETCH
 83h (131)(CD-ROM,DRFAT32) SEEK
 84h (132)(CD-ROM) PLAY AUDIO
 85h (133)(CD-ROM) STOP AUDIO
 86h (134)(CD-ROM,DRFAT32) WRITE LONG
 87h (135)(CD-ROM,DRFAT32) WRITE LONG VERIFY
 88h (136)(CD-ROM) RESUME AUDIO

Bitfields for device request status:
Bit(s)  Description (Table 02596)
 15 error
 14-11  reserved
 10 ??? set by DOS kernel on entry to some driver calls
```

```
 9  busy
 8  done (may be clear on return under European MS-DOS 4.0)
 7-0  error code if bit 15 set (see #02598)

Format of device driver request header:
Offset  Size  Description (Table 02597)
 00h  BYTE  length of request header
 01h  BYTE  subunit within device driver
 02h  BYTE  command code (see #02595)
 03h  WORD  status (filled in by device driver) (see #02596)
---DOS---
 05h  4 BYTEs reserved (unused in DOS 2.x and 3.x)
 09h  DWORD (European MS-DOS 4.0 only) pointer to next request header in
        device's request queue
    (other versions) reserved (unused in DOS 2.x and 3.x)
---STARLITE architecture---
 05h  DWORD pointer to next request header
 09h  4 BYTEs reserved
---command code 00h---
 0Dh  BYTE  (ret) number of units
 0Eh  DWORD (call) pointer to DOS device helper function (see #02599)
        (European MS-DOS 4.0 only)
    (call) pointer past end of memory available to driver (DOS 5+)
    (ret) address of first free byte following driver
 12h  DWORD (call) pointer to commandline arguments
    (ret) pointer to BPB array (block drivers) or
        0000h:0000h (character drivers)
 16h  BYTE  (DOS 3.0+) drive number for first unit of block driver (0=A)
   ---European MS-DOS 4.0---
 17h  DWORD pointer to function to save registers on stack
   ---DOS 5+ ---
 17h  WORD  (ret) error-message flag
    0001h MS-DOS should display error msg on init failure
---command code 01h---
 0Dh  BYTE  media descriptor
 0Eh  BYTE  (ret) media status
    00h don't know
    01h media has not changed
    FFh media has been changed
 0Fh  DWORD (ret, DOS 3.0+) pointer to previous volume ID if the
      OPEN/CLOSE/RM bit in device header is set and disk changed
```

```
    Note: some drives (or controllers???) forget the change line
        status if another drive is accessed afterwards. The
        DOS IO.SYS layer takes care of this by not relying
        on the reported change line status when the change
        line is not active and a different drive is accessed,
        instead it reports "don't know" to the DOS kernel.
---command code 02h---
 0Dh  BYTE  media descriptor
 0Eh  DWORD transfer address
    -> scratch sector if NON-IBM FORMAT bit in device header set
    -> first FAT sector otherwise
 12h  DWORD pointer to BPB (set by driver) (see #01663 at INT 21/AH=53h)
---command codes 03h,0Ch---
      (see also INT 21/AX=4402h"DOS 2+",INT 21/AX=4403h"DOS")
 0Dh  BYTE  media descriptor (block devices only)
 0Eh  DWORD transfer address
 12h  WORD  (call) number of bytes to read/write
    (ret) actual number of bytes read or written
---command codes 04h,08h,09h (except Compaq DOS 3.31, DR DOS 6)---
 0Dh  BYTE  media descriptor (block devices only)
 0Eh  DWORD transfer address
 12h  WORD  byte count (character devices) or sector count (block devices)
 14h  WORD  starting sector number (block devices only)
 16h  DWORD (DOS 3.0+) pointer to volume ID if error 0Fh returned
 1Ah  DWORD (DOS 4.0+) 32-bit starting sector number (block devices with
      device attribute word bit 1 set only) if starting sector
      number above is FFFFh (see INT 21/AH=52h)
---command codes 04h,08h,09h (Compaq DOS 3.31, DR DOS 6)---
 0Dh  BYTE  media descriptor (block devices only)
 0Eh  DWORD transfer address
 12h  WORD  byte count (character devices) or sector count (block devices)
 14h  DWORD 32-bit starting sector number (block devices only)
  Note: to reliably determine which variant of the request block for
      functions 04h,08h,09h has been passed to the driver, check
      the length field as well as the word at offset 14h.  If the
      length is 1Eh and 14h=FFFFh, use the DWORD at 1Ah as the
      starting sector number; if the length is 18h, use the DWORD
      at 14h; otherwise, use the WORD at 14h.
---command code 05h---
 0Dh  BYTE  byte read from device if BUSY bit clear on return
---command codes 06h,07h,0Ah,0Bh,0Dh,0Eh,0Fh---
```

```
 no further fields
---command code 10h---
 0Dh  BYTE  unused
 0Eh  DWORD transfer address
 12h  WORD  (call) number of bytes to write
    (ret) actual number of bytes written
---command codes 11h,12h---
 0Dh  BYTE  reserved
---command code 14h---
 no further fields
  Note: This is at least true for KKCFUNC.SYS' "device restore".
      KKCFUNC.SYS checks that INT 2Fh in the IVT still points
      to KKCFUNC's own INT 2Fh entry point.  In this case it
      restores the original INT 2Fh vector, as recorded at device
      init, into the IVT.
  SeeAlso: INT 2F/AH=4Dh
---command code 15h---
 no further fields
---command codes 13h,19h---
 0Dh  BYTE  category code
    00h-7Fh reserved for Microsoft
    00h unknown
    01h COMn: (serial) (DOS 3.3+)
    02h reserved for terminal control
    03h CON (DOS 3.3+)
    04h reserved for keyboard control
    05h LPTn:
    07h mouse (European MS-DOS 4.0)
    08h disk
    48h FAT32 disk control (MS-DOS 7.10+)
    80h-FFh reserved for OEM/user-defined
    9Eh (STARLITE) Media Access Control driver
    EDh (DR PalmDOS) login security
    SeeAlso: #01558
 0Eh  BYTE  function code
    00h (STARLITE) MAC Bind request
 0Fh  WORD  copy of DS at time of IOCTL call (apparently unused in DOS 3.3)
    SI contents (European MS-DOS 4.0)
 11h  WORD  offset of device driver header (see #01646)
    DI contents (European MS-DOS 4.0)
 13h  DWORD pointer to parameter block from INT 21/AX=440Ch or AX=440Dh
```

```
---command codes 80h,82h---
 0Dh  BYTE  addressing mode
    00h HSG (default)
    01h Phillips/Sony Red Book
 0Eh  DWORD transfer address (ignored for command 82h)
 12h  WORD  number of sectors to read
    (if 0 for command 82h, request is an advisory seek)
 14h  DWORD starting sector number
    logical sector number in HSG mode
    frame/second/minute/unused in Red Book mode
    (HSG sector = minute * 4500 + second * 75 + frame - 150)
 18h  BYTE  data read mode
    00h cooked (2048 bytes per frame)
    01h raw (2352 bytes per frame, including EDC/ECC)
 19h  BYTE  interleave size (number of sectors stored consecutively)
 1Ah  BYTE  interleave skip factor
    (number of sectors between consecutive portions)
---command code 83h---
 0Dh  BYTE  addressing mode
    00h HSG (default)
    01h Phillips/Sony Red Book
 0Eh  DWORD transfer address (ignored)
 12h  WORD  number of sectors to read (ignored)
 14h  DWORD starting sector number (see also above)
---command code 84h---
 0Dh  BYTE  addressing mode
    00h HSG (default)
    01h Phillips/Sony Red Book
 0Eh  DWORD starting sector number (see also above)
 12h  DWORD number of sectors to play
---command codes 85h,88h---
 no further fields
---command codes 86h,87h---
 0Dh  BYTE  addressing mode
    00h HSG (default)
    01h Phillips/Sony Red Book
 0Eh  DWORD transfer address (ignored in write mode 0)
 12h  WORD  number of sectors to write
 14h  DWORD starting sector number (also see above)
 18h  BYTE  write mode
    00h mode 0 (write all zeros)
```

```
    01h mode 1 (default) (2048 bytes per sector)
    02h mode 2 form 1 (2048 bytes per sector)
    03h mode 2 form 2 (2336 bytes per sector)
 19h  BYTE  interleave size (number of sectors stored consecutively)
 1Ah  BYTE  interleave skip factor
    (number of sectors between consecutive portions)

(Table 02598)
Values for device driver error code:
 00h  write-protect violation
 01h  unknown unit
 02h  drive not ready
 03h  unknown command
 04h  CRC error
 05h  bad drive request structure length
 06h  seek error
 07h  unknown media
 08h  sector not found
 09h  printer out of paper
 0Ah  write fault
 0Bh  read fault
 0Ch  general failure
 0Dh  reserved
 0Eh  (CD-ROM) media unavailable
 0Fh  invalid disk change

(Table 02599)
Call European MS-DOS 4.0 device helper function with:
  DL = function
       00h "SchedClock" called on each timer tick
    AL = tick interval in milliseconds
       01h "DevDone" device I/O complete
    ES:BX -> request header
    Note: must update status word first; may be called from
        an interrupt handler
       02h "PullRequest" pull next request from queue
    DS:SI -> DWORD pointer to start of device's request queue
    Return: ZF clear if pending request
          ES:BX -> request header
       ZF set if no more requests
       03h "PullParticular" remove specific request from queue
```

```
DS:SI -> DWORD pointer to start of device's request queue
ES:BX -> request header
Return: ZF set if request header not found
  04h "PushRequest" push the request onto the queue
DS:SI -> DWORD pointer to start of device's request queue
ES:BX -> request header
interrupts disabled
  05h "ConsInputFilter" keyboard input check
AX = character (high byte 00h if PC ASCII character)
Return: ZF set if character should be discarded
  ZF clear if character should be handled normally
Note: called by keyboard interrupt handler so DOS can scan
    for special input characters
  06h "SortRequest" push request in sorted order by starting sector
DS:SI -> DWORD pointer to start of device's request queue
ES:BX -> request header
interrupts disabled
  07h "SigEvent" send signal on keyboard event
AH = event identifier
Return: AL,FLAGS destroyed
  09h "ProcBlock" block on event
AX:BX = event identifier (typically a pointer)
CX = timeout in ms or 0000h for never
DH = interruptable flag (nonzero if pause may be interrupted)
interrupts disabled
Return: after corresponding ProcRun call
  CF clear if event wakeup, set if unusual wakeup
  ZF set if timeout wakeup, clear if interrupted
  AL = wakeup code, nonzero if unusual wakeup
  interrupts enabled
  BX,CX,DX destroyed
Note: block process and schedules another to run
  0Ah "ProcRun" unblock process
AX:BX = event identifier (typically a pointer)
Return: AX = number of processes awakened
  ZF set if no processes awakened
  BX,CX,DX destroyed
  0Bh "QueueInit" initialize/clear character queue
DS:BX -> character queue structure (see #02600)
Note: the queue size field must be set before calling
  0Dh "QueueWrite" put a character in the queue
```

```
    DS:BX -> character queue (see #02600)
    AL = character to append to end of queue
    Return: ZF set if queue is full
      ZF clear if character stored
      0Eh "QueueRead" get a character from the queue
    DS:BX -> character queue (see #02600)
    Return: ZF set if queue is empty
      ZF clear if characters in queue
          AL = first character in queue
      10h "GetDOSVar" return pointer to DOS variable
    AL = index of variable
        03h current process ID
    BX = index into variable if AL specifies an array
    CX = expected length of variable
    Return: CF clear if successful
          DX:AX -> variable
      CF set on error
          AX,DX destroyed
      BX,CX destroyed
    Note: the variables may not be modified
      14h "Yield" yield CPU if higher-priority task ready to run
    Return: FLAGS destroyed
      1Bh "CritEnter" begin system critical section
    DS:BX -> semaphore (6 BYTEs, initialized to zero)
    Return: AX,BX,CX,DX destroyed
      1Ch "CritLeave" end system critical section
    DS:BX -> semaphore (6 BYTEs, initialized to zero)
    Return: AX,BX,CX,DX destroyed
    Note: must be called in the context of the process which
        called CritEnter on the semaphore
Note: the DWORD pointing at the request queue must be allocated by the driver
    and initialized to 0000h:0000h.  It always points at the next request
    to be executed

Format of European MS-DOS 4.0 character queue:
Offset  Size  Description (Table 02600)
 00h  WORD  size of queue in bytes
 02h  WORD  index of next character out
 04h  WORD  count of characters in the queue
 06h  N BYTEs queue buffer
--------D-2F0803-----------------------------
```

```
INT 2F U - DOS 4.0+ DRIVER.SYS support - GET DRIVE DATA TABLE LIST
  AX = 0803h
Return: DS:DI -> first drive data table in list (see #02601,#02602,#02603)
Note: not available under DR DOS 5.0, but supported by Novell DOS 7 (using
    the MS-DOS 4+ data table format)
SeeAlso: AX=0801h

Format of DOS 3.30 drive data table:
Offset  Size  Description (Table 02601)
 00h  DWORD pointer to next table (offset FFFFh if last table)
 04h  BYTE  physical unit number (for INT 13)
 05h  BYTE  logical drive number (0=A:)
 06h 19 BYTEs BIOS Parameter Block (see also INT 21/AH=53h)
    Offset  Size  Description
     00h  WORD  bytes per sector
     02h  BYTE  sectors per cluster, FFh if unknown
     03h  WORD  number of reserved sectors
     05h  BYTE  number of FATs
     06h  WORD  number of root dir entries
     08h  WORD  total sectors
     0Ah  BYTE  media descriptor, 00h if unknown
     0Bh  WORD  sectors per FAT
     0Dh  WORD  sectors per track
     0Fh  WORD  number of heads
     11h  WORD  number of hidden sectors
 19h  BYTE  flags
    bit 6: 16-bit FAT instead of 12-bit FAT
 1Ah  WORD  number of DEVICE OPEN calls without corresponding DEVICE CLOSE
 1Ch 11 BYTEs volume label or "NO NAME    " if none (always "NO NAME" for
      fixed media)
 27h  BYTE  terminating null for volume label???
 28h  BYTE  device type (see #01561 at INT 21/AX=440Dh"DOS 3.2+")
 29h  WORD  bit flags describing drive (see #02604)
 2Bh  WORD  number of cylinders
 2Dh 19 BYTEs BIOS Parameter Block for highest capacity supported
 40h  3 BYTEs ???
 43h  9 BYTEs filesystem type???, default = "NO NAME  "
    (apparently only MS-DOS 3.30 fixed media, nulls for removable
      media and PC-DOS 3.30)
 4Ch  BYTE  least-significant byte of last-accessed cylinder number
---removable media---
```

```
 4Dh  DWORD time of last access in clock ticks (FFFFFFFFh if never)
---fixed media---
 4Dh  WORD  partition (FFFFh = primary, 0001h = extended)
 4Fh  WORD  absolute cylinder number of partition's start on physical
      drive (always FFFFh if primary partition)
SeeAlso: #02602,#02603

Format of COMPAQ DOS 3.31 drive data table:
Offset  Size  Description (Table 02602)
 00h  DWORD pointer to next table (offset FFFFh if last table)
 04h  BYTE  physical unit number (for INT 13)
 05h  BYTE  logical drive number (0=A:)
 06h 25 BYTEs BIOS Parameter Block (see #02603)
 1Fh  6 BYTEs reserved fields from BPB above???
 25h  BYTE  flags
    bit 6: 16-bit FAT instead of 12-bit FAT
    bit 5: large volume???
 26h  WORD  device-open count???
 28h 11 BYTEs volume label or "NO NAME    " if none (always "NO NAME" for
      fixed media)
 33h  BYTE  terminating null for volume label
 34h  BYTE  device type (see #01561 at INT 21/AX=440Dh"DOS 3.2+")
 35h  WORD  bit flags describing drive (see #02604)
 37h  WORD  number of cylinders
 39h 25 BYTEs BIOS parameter block for highest capacity drive supports
 52h  6 BYTEs ??? apparently always zeros
 58h  BYTE  least-significant byte of last-accessed cylinder number
---removable media---
 59h  DWORD time of last access in clock ticks (FFFFFFFFh if never)
---fixed media---
 59h  WORD  partition (FFFFh = primary, 0001h = extended)
 5Bh  WORD  absolute cylinder number of partition's start on physical
      drive (always FFFFh if primary partition)
SeeAlso: #02601,#02603

Format of DOS 4.0-7.0 drive data table:
Offset  Size  Description (Table 02603)
 00h  DWORD pointer to next table (offset FFFFh if last table)
 04h  BYTE  physical unit number (for INT 13)
 05h  BYTE  logical drive number (0=A:)
 06h 25 BYTEs BIOS Parameter Block (see also INT 21/AH=53h)
```

```
    Offset  Size  Description
     00h  WORD  bytes per sector
     02h  BYTE  sectors per cluster, FFh if unknown
     03h  WORD  number of reserved sectors
     05h  BYTE  number of FATs
     06h  WORD  number of root dir entries
     08h  WORD  total sectors (refer to offset 15h if zero)
     0Ah  BYTE  media descriptor, 00h if unknown
     0Bh  WORD  sectors per FAT
     0Dh  WORD  sectors per track
     0Fh  WORD  number of heads
     11h  DWORD number of hidden sectors
     15h  DWORD total sectors if WORD at 08h is zero
 1Fh  BYTE  flags
    bit 6: 16-bit FAT instead of 12-bit
    bit 7: unsupportable disk (all accesses will return Not Ready)
 20h  WORD  device-open count
 22h  BYTE  device type (see #01561 at INT 21/AX=440Dh"DOS 3.2+")
 23h  WORD  bit flags describing drive (see #02604)
 25h  WORD  number of cylinders (for partition only, if hard disk)
 27h 25 BYTEs BIOS Parameter Block for default (highest) capacity supported
 40h  6 BYTEs reserved (part of BPB above)
 46h  BYTE  last track accessed
---removable media---
 47h  DWORD time of last access in clock ticks (FFFFFFFFh if never)
---fixed media---
 47h  WORD  partition (FFFFh = primary, 0001h = extended)
    always 0001h for DOS 5+
 49h  WORD  absolute cylinder number of partition's start on physical drive
    (FFFFh if primary partition in DOS 4.x)
------
 4Bh 11 BYTEs volume label or "NO NAME    " if none (apparently taken from
      extended boot record rather than root directory)
 56h  BYTE  terminating null for volume label
 57h  DWORD serial number
 5Bh  8 BYTEs filesystem type ("FAT12   " or "FAT16 ")
 63h  BYTE  terminating null for filesystem type
SeeAlso: #02601,#02602

Bitfields for flags describing drive:
Bit(s)  Description (Table 02604)
```

```
 0  fixed media
 1  door lock ("changeline") supported
 2  current BPB locked
 3  all sectors in a track are the same size
 4  physical drive has multiple logical units
 5  current logical drive for shared physical drive
 6  disk change detected
 7  device parameters were changed (set DASD before formatting)
  (see #01560 at INT 21/AX=440Dh"DOS 3.2+")
 8  disk reformatted (BPB of current media was changed)
 9  access flag (fixed media only, disables reads and writes)
  (see #01566 at INT 21/AX=440Dh"DOS 3.2+")
--------f-2F1000----------------------------
INT 2F - SHARE - INSTALLATION CHECK
  AX = 1000h
Return: AL = status
      00h not installed, OK to install
      01h not installed, not OK to install
      FFh installed
Notes:  supported by OS/2 v1.3+ compatibility box, which always returns AL=FFh
  if DOS 4.01 SHARE was automatically loaded, file sharing is in an
    inactive state (due to the undocumented /NC flag used by the autoload
    code) until this call is made
  DR DOS 5.0 SHARE 1.00 only checks for AH=10h and in this case does not
    chain to a previous handler, DR DOS 6.0 SHARE 1.02+ properly chains
    for any values other than AX=1000h and the private FDOS hook
    AX=1001h.  However, under DR PalmDOS and Novell DOS 7+, the
    SHARE 2.00+ only tests for the AX=1000h install check, since it no
    longer uses AX=1001h to hook into the system.
  DOS 5+ chains to the previous handler if AL <> 00h on entry
  Windows Enhanced mode hooks this call and reports that SHARE is
    installed even when it is not
BUGS: values of AL other than 00h put DOS 3.x SHARE into an infinite loop
    (08E9: OR  AL,AL
     08EB: JNZ 08EB) <- the buggy instruction (DOS 3.3)
  values of AL other than described here put PC-DOS 4.00 into the same
    loop (the buggy instructions are the same)
SeeAlso: AX=1080h,INT 21/AH=52h,INT 21/AX=4457h/DX=FFFFh
--------d-2F1001----------------------------
INT 2F U - DR DOS 6.0+ FDOS EXTENSIONS - INSTALL FDOS HOOK (SHARE / DELWATCH)
  AX = 1001h
```

```
  DX:BX -> new FDOS stub entry function
Return: nothing
Notes:  the default handler for the pointer set by this call under DELWATCH
    simply returns with CF set
  In Digital Research terminology FDOS is the part of the BDOS kernel
    (IBMDOS.COM) responsible for the file system and related tasks, and
    its functionality is also used by the BDOS kernel itself. However,
    for reasons of performance and code size, it uses direct calling.
  This interrupt allows external components like SHARE file locking or
    DELWATCH delete tracking software to hook into the internal backdoor
    INT 2F/AH=10h/AL<=09h chain in the kernel, so that they can actually
    allows the kernel to ensure that several conditions are met when
    passing control to these registered components including proper
    maintaining A20, re-enterancy, or critical section mutexing
    (e.g. getting the disk sub-system queue "MXdisk").
  The default handler in the BDOS just sets the carry flag and returns.
  Currently known clients to this shared interface are DR DOS 6.0
    SHARE 1.xx, DR DOS 6.0+ DELWATCH 1.00+, which both chain onto
    the same call far address, and the version of AddStor's SuperStor
    which is bundled with DR DOS 6.0
  However, the DR PalmDOS, and Novell DOS 7 - DR-DOS 7.03 SHARE 2.00-2.05
    do not use this function to hook into the system. Instead they fix up
    the share stubs directly. DR PalmDOS SHARE 2.00 uses the stubs at
    Table !!! INT 21/AH=52h, while Novell DOS 7 - DR-DOS 7.03
    SHARE 2.01-2.05 use the private set of share stubs at Table !!! at
    INT 21/AX=4458h).  Future releases may possibly again fix up the
    share stubs at INT 21/AH=52h.
SeeAlso: AX=1000h,AX=F800h
--------D-2F1002CHFF------------------------
INT 2F CU - Novell DOS 7+ FDOS EXTENSIONS - READ BUFFERS
  AX = 1002h
  CH = FFh (pre-read required)
  CL = buffer type (here FAT, DIR, or DATA, see below)
  AH:DX = 24 bit sector number
Return: ES:SI -> buffer control block
Notes:  This private function is called internally by the OS kernel to
    ask the FDOS to find the corresponding buffer.  By using the
    INT 2F/AX=1001h FDOS hook, the call can be intercepted by external
    system components such as the DELWATCH TSR.
  This function must under no circumstances be called by applications!
SeeAlso: INT 2F/AX=1001h,AX=1003h,AX=1008h,AX=1009h,AX=10FEh
```

```
Bitfields for Novell DOS 7 FDOS buffer type:
Bit(s)  Description (Table 04099)
 7  remote (on a network drive)
 6  dirty (modified)
 3  data sector
 2  directory sector
 1  FAT sector
--------D-2F1003-----------------------------
INT 2F CU - Novell DOS 7+ FDOS EXTENSIONS - FLUSH BUFFERS
  AX = 1003h
  BL = drive???
  BH = buffer type to flush (BF_ISFAT+BF_ISDIR+BF_ISDATA)
Return: nothing???
Notes:  This private function is called internally by the OS kernel to
    ask the FDOS to flush all buffer.  By using the INT 2F/AX=1001h FDOS
    hook, the call can be intercepted by external system components such
    as the DELWATCH TSR.
  This function must under no circumstances be called by applications!
SeeAlso: INT 2F/AX=1001h,AX=1004h,AX=10FEh
--------D-2F1004-----------------------------
INT 2F CU - Novell DOS 7+ FDOS EXTENSIONS - FREE FAT CHAIN
  AX = 1004h
  BX = first block to release on current drive
Return: nothing
Notes:  This private function is called internally by the OS kernel to
    ask the FDOS to release the FAT chain.  By using the INT 2F/AX=1001h
    FDOS hook, the call can be intercepted by external system components
    such as the DELWATCH TSR.
  This function must under no circumstances be called by applications!
SeeAlso: INT 2F/AX=1001h,AX=1005h,AX=10FEh
--------D-2F1005-----------------------------
INT 2F CU - Novell DOS 7+ FDOS EXTENSIONS - ALLOCATE CLUSTER
  AX = 1005h
  BX = block from which to start search (e.g. current end of file)
       0000h = start of disk
Return: AX or BX??? = allocated cluster (already marked as End of Chain)
      or 0000h if none available
Notes:  This private function is called internally by the OS kernel to
    ask the FDOS to allocate disk space.  By using the
    INT 2F/AX=1001h FDOS hook, the call can be intercepted by external
```

```
    system components such as the DELWATCH TSR.
  This function must under no circumstances be called by applications!
SeeAlso: INT 2F/AX=1001h,AX=1004h,AX=1007h,AX=10FEh
--------D-2F1006-----------------------------
INT 2F CU - Novell DOS 7+ FDOS EXTENSIONS - NEXT CLUSTER / READ FAT???
  AX = 1006h
  BX = current cluster number
Return: AX or BX??? = next cluster in chain
Notes:  This private function is called internally by the OS kernel to
    ask the FDOS to get the next cluster in a file.  By using the
    INT 2F/AX=1001h FDOS hook, the call can be intercepted by external
    system components such as the DELWATCH TSR.
  This function must under no circumstances be called by applications!
SeeAlso: INT 2F/AX=1001h,AX=1005h,AX=1007h,AX=10FEh
--------D-2F1007-----------------------------
INT 2F CU - Novell DOS 7+ FDOS EXTENSIONS - UPDATE FAT ENTRY / WRITE FAT???
  AX = 1007h
  BX = FAT entry to change
  DX = new value
Return: nothing
Notes:  This private function is called internally by the OS kernel to
    ask the FDOS to update the FAT.  By using the INT 2F/AX=1001h FDOS
    hook, the call can be intercepted by external system components such
    as the DELWATCH TSR.
  This function must under no circumstances be called by applications!
SeeAlso: INT 2F/AX=1001h,AX=1005h,AX=1006h,AX=1008h,AX=10FEh
--------D-2F1008-----------------------------
INT 2F CU - Novell DOS 7+ FDOS EXTENSIONS - FIXUP CHECKSUMS / DIR UPDATE???
  AX = 1008h
  BX = segment of directory buffer
  CX = cluster to fixup (0 = root)
  DI = directory entry index (truncated to cluster if subdirectory)
  BX:SI -> directory entry (single entry for hashing)
Return: nothing
Notes:  This private function is called internally by the OS kernel to
    ask the FDOS to fixup hashing/checksums.  By using the
    INT 2F/AX=1001h FDOS hook, the call can be intercepted by external
    system components such as the DELWATCH TSR.
  This function must under no circumstances be called by applications!
SeeAlso: INT 2F/AX=1001h,AX=1007h,AX=1009h,AX=10FEh
--------D-2F1009-----------------------------
```

```
INT 2F CU - Novell DOS 7+ FDOS EXTENSIONS - DIRECTORY BUFFER INFO
  AX = 1009h
Return: ES:DI -> 128-byte directory record buffer
  ES:SI -> directory BCB structure (see #04101)
Notes:  This private function is called internally by the OS kernel.  By using
    the INT 2F/AX=1001h FDOS hook, the call can be intercepted by
    external system components such as the DELWATCH TSR.
  This function must under no circumstances be called by applications!
SeeAlso: INT 2F/AX=1001h,AX=10FEh

Format of Novell DOS 7+ FDOS directory BCB:
Offset  Size  Description (Table 04101)
 00h  BYTE  drive (FFh = invalid)
 01h  BYTE  low byte of record number
 02h  BYTE  middle byte of record number
 03h  BYTE  high byte of record number
--------d-2F1010CX0000-----------------------
INT 2F CU - Novell DOS 7+ FDOS EXTENSIONS - SUPERSTOR - QUERY PHYS FREE SPACE
  AX = 1010h
  CX = 0000h
Return: CX = free space
      0000h if no physical space left on the drive
      else there is still space available
Notes:  This private function is internally called by the FDOS part of the
    OS kernel on "out of disk space" conditions.  It allows optimized
    behaviour of DELWATCH delete tracking software in conjunction with
    disk compression, and was implemented for SuperStor (but it would
    also work with other disk compression if they hook this function).
    If there truly is no physical space left on the drive, the FDOS asks
    DELWATCH to purge files in its queue until enough space has been
    freed, or there actually is no free disk space any more.
  By using the INT 2F/AX=1001h FDOS hook, the call can be intercepted by
    external system components such as SuperStor, bundled with DR DOS
    "Panther" and Novell DOS 7 BETAs.
  This function must under no circumstances be called by applications!
SeeAlso: INT 2F/AX=1001h,AX=10FEh
--------y-2F1020----------------------------
INT 2F CU - Novell DOS 7+ FDOS EXTENSIONS - CREATE PASSWORD ENTRY
  AX = 1020h
  ???
Return: ???
```

```
Notes:  This private function is called internally by the OS kernel to an
    optional SECURITY TSR.  By using the INT 2F/AX=1001h FDOS hook, it
    can be intercepted by external system components such as
    Multiuser SECURITY, bundled with DR DOS "Panther" Beta 1.
  This function must under no circumstances be called by applications!
SeeAlso: INT 2F/AX=1001h,AX=1021h,AX=1022h,AX=10FEh
--------y-2F1021-----------------------------
INT 2F CU - Novell DOS 7+ FDOS EXTENSIONS - CHANGE PASSWORD ENTRY
  AX = 1021h
  CL > 05h ???
  BX -> matching directory entry???
  AL = directory attributes???
Return: ???
  CF set on error (password change not allowed)
Notes:  This private function is called internally by the OS kernel to an
    optional SECURITY TSR.  By using the INT 2F/AX=1001h FDOS hook, it
    can be intercepted by external system components such as
    Multiuser SECURITY, bundled with DR DOS "Panther" Beta 1.
  This function must under no circumstances be called by applications!
SeeAlso: INT 2F/AX=1001h,AX=1020h,AX=1022h,AX=10FEh
--------y-2F1022-----------------------------
INT 2F CU - Novell DOS 7+ FDOS EXTENSIONS - CHECK PASSWORD ENTRY
  AX = 1022h
  ???
Return: ???
Notes:  This private function is called internally by the OS kernel to an
    optional SECURITY TSR.  By using the INT 2F/AX=1001h FDOS hook, it
    can be intercepted by external system components such as
    Multiuser SECURITY, bundled with DR DOS "Panther" Beta 1.
  This function must under no circumstances be called by applications!
SeeAlso: INT 2F/AX=1001h,AX=1020h,AX=1021h,AX=10FEh
--------f-2F1040-----------------------------
INT 2F U - DOS 4 only SHARE internal - ???
  AX = 1040h
  ???
Return: AL = FFh???
SeeAlso: AX=1000h
--------f-2F1080-----------------------------
INT 2F U - DOS 4 only SHARE internal - TURN ON FILE SHARING CHECKS
  AX = 1080h
Return: AL = status
```

```
      F0h successful
      FFh checking was already on
Note: DOS 4.x SHARE has dual functions: FCB support for large (>32M) media
    and file sharing checks.  The undocumented commandline flag /NC can
    be used to disable the sharing code.
SeeAlso: AX=1000h,AX=1081h
--------f-2F1081-----------------------------
INT 2F U - DOS 4 only SHARE internal - TURN OFF FILE SHARING CHECKS
  AX = 1081h
Return: AL = status
      F0h successful
      FFh checking was already off
Note: (see AX=1080h)
SeeAlso: AX=1000h,AX=1080h
--------O-2F10FE-----------------------------
INT 2F U - DR DOS 6.0+ DELWATCH.EXE - INSTALLATION CHECK
  AX = 10FEh
Return: AL = FFh if installed and active
      AH = internal version number
          10h for DR DOS 6.0 DELWATCH 1.0 / 1.1 (through 1993/03/19)
    20h for Novell DOS 7+ DELWATCH 2.0+
      DX:BX -> private entry point (see #02605)
Notes:  The DR DOS 6.0 DELWATCH 1.x used to store information about deleted
    files in a hidden file named @DLWATCH.DAT, however the Novell DOS 7+
    DELWATCH 2.0+ stores all the info in previously unused fields in
    the files' directory entries. (See table !!! at INT 21h/11h for
    details). This, however, now causes problems on systems also running
    Windows 9x since Microsoft decided to use a rather similar but
    incompatible method to store long filenames etc. in these entries.
  Running DELWATCH 2.x on a system which previously used DELWATCH 1.x,
    the @DLWATCH.DAT file will be abandoned and converted to the new
    method.
SeeAlso: #01352,INT 21/AX=4306h,INT 21/AX=5704h,INT 2F/AX=1001h

(Table 02605)
Call DELWATCH private entry point with:
  AH = function
      00h (OS hook) installation check
    AL = 00h required for DELWATCH 1.x
    Return: CF clear
      AX = 0000h
```

```
    CX = 0004h (unsupported function)
    01h (DELWATCH 1.x) New Disk
        ???
    01h (DELWATCH 2.0+) disable DELWATCH on drive
AL = drive number (00h = A:)
Return: AX = status (0000h if failed, FFFFh if successful)
    02h (OS hook) Delete File
AL = drive number (00h = A:)
DX = directory cluster number (0000h for root directory)
CX = directory entry number
DS:BX -> directory entry
ES,DS must be valid selectors if called in protected mode
Return: DS:BX -> updated directory entry
    CF set if file is to be deleted by the OS
    CF clear if DELWATCH has placed the file in its queue
Note: deletes the directory entry
    03h (OS hook) Free Clusters
AL = drive number (00h = A:)
CX = number of clusters currently free (do not free if > 1)
DX = preferred 'search from cluster' (ignored by DELWATCH 2.0)
Return: CF clear if clusters freed
    CF set if no clusters freed
    DX = new 'search from' cluster (one before first free)
    04h (OS hook) free root directory entry
AL = drive number (00h = A:)
Return: CF set if no directory entry freed
    05h (OS hook) return free space
AL = drive number (00h = A:)
CX = number of free clusters
Return: CX = updated number of free clusters
Notes:  adds space used by "deleted" files to free space
    call is chained
    06h enable DELWATCH on drive
AL = drive number with bit 7 set (80h = A:, etc.)
    (DELWATCH 2.0+: set bit 6 for removable drives)
BX = maximum files of same name in one directory to save
CX = maximum files to save on this disk
DS:DX -> MEMDESC??? for drive data (see #04104)
DS:SI -> MEMDESC??? for DWLIST (see #04104)
ES,DS must contain valid selectors if called in protected mode
Return: AX = status
```

```
        0000h failed
        FFFFh successful
    CX = error code on failure (see #04102)
        (0004h "wrong version" if AL < 80h on entry)
    07h (DELWATCH 1.x) disable DELWATCH on drive
???
    07h (DELWATCH 2.0+) (OS hook) new disk
AL = drive (00h = A:, etc.)
ES:BX -> DOS DDSC structure
CF set if not enabled
Return: ???
    08h set file extensions list
AL = sense (00h exclude named extensions, 01h only named ext.)
DS:BX -> 31-byte ASCIZ extension list (three blank-padded bytes
      per extension)
Return: AX = FFFFh (successful)
    09h adjust pending delete space
AL = drive number (00h = A:)
CX = number of clusters being freed
Return: AX = 0000h if drive not enabled
    0Ah remove DELWATCH entry
AL = drive number (00h = A:)
DX = directory cluster number (0000h if root directory)
CX = directory entry number
BX:SI -> filename
ES,DS must contain valid selectors if called in protected mode
Return: AX > 0000h if entry found in DWLIST
    0Bh enable NEWDISK
Return: AX > 0000h if successful (FFFFh for DELWATCH 2.0)
see also function 0Dh
    0Ch (DELWATCH 1.x) drive status
AL = drive number (00h = A:, etc.)
Return: AX = drive data segment, 0000h if not enabled
    CX = pending delete space, if drive enabled
    0Ch (DELWATCH 2.0+) check if drive enabled
AL = drive number with bit 7 set (80h = A:, etc.)
DS:DX -> MEMDESC for drive data (see #04104)
    (DX = 0000h if not required)
DS:SI -> MEMDESC for DWLIST (see #04104)
    (SI = 0000h if not required)
ES,DS must contain valid selectors if called in protected mode
```

```
    Return: AX = drive status (see also #04103)
          0000h disabled or error
        CX = error code (see #04102)
          0001h drive enabled
        CX = pending delete space, FFFFh if NEWDISK
              not yet called
      0Dh disable NEWDISK
    BX = segment address of bitmap buffer
    Return: AX > 0000h if successful (FFFFh for DELWATCH 2.0)
    see also function 0Bh
      0Eh (DELWATCH 2.0+) (OS hook) purge file
    AL = drive number (00h = A:)
    DX = directory cluster number (0000h if root directory)
    CX = directory entry number
    Return: CF set if drive not enabled
      CF clear
          AX = status
        0000h successfully purged
        else error code (see #04102)
      0Fh (DELWATCH 2.0+) (OS hook) undelete file
    AL = drive number (00h = A:)
    DX = directory cluster number (0000h if root directory)
    CX = directory entry number
    Return: CF set if drive not enabled
      CF clear
          AX = status
        0000h successfully undeleted
        else error code (see #04102)
Return: AX = 0000h, CX = 0001h (see #04102) if DELWATCH busy
  registers unchanged if AH > 0Fh on entry
Notes:  functions marked "OS hook" must under no circumstances by called by
    external applications, as this would bypass the serialization
    performed by the kernel and cause problems at least in multitasking
    environments.
  two functions have been swapped between DELWATCH 1.x and DELWATCH 2.0
    to ensure that DELWATCH 1.x calls will not do anything under newer
    versions of the OS; for the same reason, the drive number in AL
    sometimes requires that bit 7 be set for DELWATCH 2.0+.
SeeAlso: AX=1001h,AX=1010h

(Table 04102)
```

```
Values for DELWATCH error codes:
 0001h  reentered (DELWATCH busy)
 0002h  not enabled
 0003h  not found
 0004h  wrong version of DELWATCH
 0005h  memory allocation
SeeAlso: #04103

(Table 04103)
Values for DELWATCH drive status:
 0000h  drive not enabled
 0001h  OK
 0002h  no bitmap
 0003h  zero files
 0004h  cross-linked files
SeeAlso: #04102

Format of DELWATCH MEMDESC structure:
Offset  Size  Description (Table 04104)
 00h  BYTE  memory type
    01h protected mode (DPMS)
    02h XMS
    03h upper (high) memory
    04h low memory
 01h  DWORD location
    (conventional memory) WORD: segment base address
    (XMS) WORD: XMS handle
    (DPMS) DWORD: DPMS memory 32-bit base address
 05h  DWORD length in bytes
 09h  DWORD allocation
    (conventional memory) WORD: memory block segment
    (XMS) WORD: XMS handle (same as location handle)
    (DPMS) DWORD: DPMS 32-bit handle
--------O-2F10FF-----------------------------
INT 2F U - DR DOS 5.0 - FIXUP SHARE STUB TABLE???
  AX = 10FFh
  ES:BX -> new SHARE stub table to use???
Return: DS destroyed???
Notes:  Sets a pointer in the kernel. ES:BX points to a structure in SHARE's
    segment, which presumeably contains a number (4 or 11???) of entries
    of 5 bytes each and is probably part of some kind of share stub
```

```
    dispatcher that gets fixed up by this call.
  This was seen called by DR DOS 5.0 SHARE 1.00 (when INT 2F/AX=1000h
    revealed that SHARE was not installed) before it hooked INT 2Fh
    via INT 21/AH=35h to provide its install check function. It preserved
    the DS register before calling.
  However, the DR DOS 6.0+ SHARE 1.02+ uses INT 2F/AX=1001h to hook
    into the OS, DR PalmDOS SHARE 2.00 directly fixes up the share stubs
    in table !!! at INT 21h/52h, and Novell DOS 7 - DR-DOS 7.03
    SHARE 2.01-2.05 maintains the private set of share stubs in
    Table !!! at INT 21/AX=4458h).
  This function was probably used between 1990/02/09 and 1991/03/15.
SeeAlso: INT 2F/AX=1001h
--------N-2F1100-----------------------------
INT 2F C - NETWORK REDIRECTOR - INSTALLATION CHECK
  AX = 1100h
Return: AL = status
      00h not installed, OK to install
      01h not installed, not OK to install
      FFh installed
    AH = product identifier (ad hoc by various manufacturers)
        00h if PC Tools v8 DRIVEMAP
        42h ('B') for Beame&Whiteside BWNFS v3.0a
        6Eh ('n') for NetWare Lite v1.1 CLIENT
Notes:  this function is called by the DOS 3.1+ kernel
  in DOS 4.x only, the 11xx calls are all in IFSFUNC.EXE, not in the
    PC LAN Program redirector; DOS 5+ moves the calls back into the
    redirector
  the PC Network 1.00 redirector (renamed to PC LAN Program in 1.1-1.3)
    only supports AL=00h-27h
--------d-2F1100SFDADA-----------------------
INT 2F - MSCDEX (MS CD-ROM Extensions) - INSTALLATION CHECK
  AX = 1100h subfn DADAh
  STACK: WORD DADAh
Return: AL = status
      00h not installed, OK to install
    STACK unchanged
      01h not installed, not OK to install
    STACK unchanged
      FFh installed
    STACK: WORD ADADh if MSCDEX installed
        DADBh if Lotus CD/Networker installed
```

```
Note: although MSCDEX sets the stack word to ADADh on return, any value other
    than DADAh is considered to mean that MSCDEX is already installed;
    Lotus CD/Networker v4+ uses this feature to fool MSCDEX into
    thinking it is already installed when it is in fact CD/Networker
    that is installed
Index:  installation check;Lotus CD/Networker
Index:  Lotus CD/Networker;installation check
--------N-2F1101-----------------------------
INT 2F CU - NETWORK REDIRECTOR - REMOVE REMOTE DIRECTORY
  AX = 1101h
  SS = DOS DS
  SDA first filename pointer -> fully-qualified directory name
  SDA CDS pointer -> current directory structure for drive with dir
Return: CF set on error
      AX = DOS error code (see #01680 at INT 21/AH=59h/BX=0000h)
  CF clear if successful
Note: this function is called by the DOS 3.1+ kernel
SeeAlso: AX=1103h,AX=1105h,INT 21/AH=3Ah,INT 21/AH=60h
--------N-2F1102-----------------------------
INT 2F CU - IFSFUNC.EXE (DOS 4.x only) - REMOVE REMOTE DIRECTORY
  AX = 1102h
  SS = DOS DS
  SDA first filename pointer -> fully-qualified directory name
  SDA CDS pointer -> current directory structure for drive with dir
Return: CF set on error
      AX = DOS error code (see #01680 at INT 21/AH=59h/BX=0000h)
  CF clear if successful
Note: appears to be identical to AX=1101h; MS internal documentation calls
    this function "SEQ_RMDIR"
SeeAlso: AX=1101h
--------N-2F1103-----------------------------
INT 2F CU - NETWORK REDIRECTOR - MAKE REMOTE DIRECTORY
  AX = 1103h
  SS = DOS DS
  SDA first filename pointer -> fully-qualified directory name
  SDA CDS pointer -> current directory structure for drive with dir
Return: CF set on error
      AX = DOS error code (see #01680 at INT 21/AH=59h/BX=0000h)
  CF clear if successful
Note: this function is called by the DOS 3.1+ kernel
SeeAlso: AX=1101h,AX=1105h,INT 21/AH=39h,INT 21/AH=60h
```

```
--------N-2F1104-----------------------------
INT 2F CU - IFSFUNC.EXE (DOS 4.x only) - MAKE REMOTE DIRECTORY
  AX = 1104h
  SS = DOS DS
  SDA first filename pointer -> fully-qualified directory name
  SDA CDS pointer -> current directory structure for drive with dir
Return: CF set on error
      AX = DOS error code (see #01680 at INT 21/AH=59h/BX=0000h)
  CF clear if successful
Note: appears to be identical to AX=1103h
SeeAlso: AX=1103h
--------N-2F1105-----------------------------
INT 2F CU - NETWORK REDIRECTOR - CHDIR
  AX = 1105h
  SS = DOS DS
  SDA first filename pointer -> fully-qualified directory name
  SDA CDS pointer -> current directory structure for drive with dir
Return: CF set on error
      AX = DOS error code (see #01680 at INT 21/AH=59h/BX=0000h)
  CF clear if successful
      CDS updated with new path
Notes:  this function is called by the DOS 3.1+ kernel
  directory string in CDS should not have a terminating backslash unless
    the current directory is the root
SeeAlso: AX=1101h,AX=1103h,INT 21/AH=3Bh,INT 21/AH=60h
--------N-2F1106-----------------------------
INT 2F CU - NETWORK REDIRECTOR - CLOSE REMOTE FILE
  AX = 1106h
  ES:DI -> filled-in SFT (assumed to point at SDA's current SFT field)
Return: CF set on error
      AX = DOS error code (see #01680 at INT 21/AH=59h/BX=0000h)
  CF clear if successful
      SFT updated (redirector must decrement open count, which may be
      done with INT 2F/AX=1208h)
  ES:DI must be preserved
Note: this function is called by the DOS 3.1+ kernel
SeeAlso: AX=1116h,AX=1201h,AX=1208h,AX=1227h,INT 21/AH=3Eh
--------N-2F1107-----------------------------
INT 2F CU - NETWORK REDIRECTOR - COMMIT REMOTE FILE
  AX = 1107h
  ES:DI -> filled-in SFT (assumed to point at SDA's current SFT field)
```

```
Return: CF set on error
      AX = DOS error code (see #01680 at INT 21/AH=59h/BX=0000h)
  CF clear if successful
      all buffers for file flushed
      directory entry updated
  ES:DI must be preserved
Desc: perform all the buffer flushing, directory updates, etc. that would be
    performed on a file close, but do not decrement the SFT reference
    count
Note: this function is called by the DOS 3.1+ kernel
SeeAlso: INT 21/AH=68h,INT 21/AX=5D01h
--------N-2F1108-----------------------------
INT 2F CU - NETWORK REDIRECTOR - READ FROM REMOTE FILE
  AX = 1108h
  ES:DI -> SFT
      SFT DPB field -> DPB of drive containing file
  CX = number of bytes
  SS = DOS DS
  SDA DTA field -> user buffer
Return: CF set on error
      AX = DOS error code (see #01680 at INT 21/AH=59h/BX=0000h)
  CF clear if successful
      CX = number of bytes read (0000h = end of file)
      SFT updated
Note: this function is called by the DOS 3.1+ kernel
SeeAlso: AX=1109h,AX=1229h,INT 21/AH=3Fh"DOS",INT 21/AX=5D06h
--------N-2F1109-----------------------------
INT 2F CU - NETWORK REDIRECTOR - WRITE TO REMOTE FILE
  AX = 1109h
  ES:DI -> SFT
      SFT DPB field -> DPB of drive containing file
  CX = number of bytes
  SS = DOS DS
  SDA DTA field -> user buffer
Return: CF set on error
      AX = DOS error code (see #01680 at INT 21/AH=59h/BX=0000h)
  CF clear if successful
      CX = number of bytes written
      SFT updated
Notes:  this function is called by the DOS 3.1+ kernel
  PrintCache v3.1 PCACHE.EXE intercepts this function for SFTs where
```

```
    the Device Driver Header field points at PCACHE, but does not
    intercept any other network redirector functions
SeeAlso: AX=1107h,AX=1108h,INT 21/AH=40h,INT 21/AX=5D06h
--------N-2F110A-----------------------------
INT 2F CU - NETWORK REDIRECTOR (DOS 3.x only) - LOCK REGION OF FILE
  AX = 110Ah
  BX = file handle
  CX:DX = starting offset
  SI = high word of size
  STACK: WORD low word of size
  ES:DI -> SFT
      SFT DPB field -> DPB of drive containing file
  SS = DOS DS
Return: CF set on error
     AL = DOS error code (see #01680 at INT 21/AH=59h/BX=0000h)
  STACK unchanged
Notes:  this function is called by the DOS 3.10-3.31 kernel
  the redirector is expected to resolve lock conflicts
SeeAlso: AX=110Bh,INT 21/AH=5Ch
--------N-2F110A-----------------------------
INT 2F CU - NETWORK REDIRECTOR (DOS 4.0+) - LOCK/UNLOCK REGION OF FILE
  AX = 110Ah
  BL = function
      00h lock
      01h unlock
  CX = number of lock/unlock parameters (0001h for DOS 4.0-6.1)
  DS:DX -> parameter block (see #02606)
  ES:DI -> SFT
      SFT DPB field -> DPB of drive containing file
  SS = DOS DS
Return: CF set on error
     AL = DOS error code (see #01680 at INT 21/AH=59h/BX=0000h)
Notes:  this function is called by the DOS 4.0+ kernel
  the redirector is expected to resolve lock conflicts
SeeAlso: AX=110Bh,INT 21/AH=5Ch

Format of parameter block entry [array, but currently limited to single entry]:
Offset  Size  Description (Table 02606)
 00h  DWORD start offset
 04h  DWORD size of region
--------N-2F110B-----------------------------
```

```
INT 2F CU - NETWORK REDIRECTOR (DOS 3.x only) - UNLOCK REGION OF FILE
  AX = 110Bh
  BX = file handle
  CX:DX = starting offset
  SI = high word of size
  STACK: WORD low word of size
  ES:DI -> SFT for file
       SFT DPB field -> DPB of drive containing file
Return: CF set on error
     AL = DOS error code (see #01680 at INT 21/AH=59h/BX=0000h)
  STACK unchanged
Note: this function is called by the DOS 3.1-3.31 kernel; DOS 4.0+ calls
    AX=110Ah instead
SeeAlso: AX=110Ah,INT 21/AH=5Ch
--------N-2F110C-----------------------------
INT 2F CU - NETWORK REDIRECTOR - GET DISK INFORMATION
  AX = 110Ch
  ES:DI -> current directory structure for desired drive
Return: CF clear if data valid
       AL = sectors per cluster
       AH = media ID byte
       BX = total clusters
       CX = bytes per sector
       DX = number of available clusters
  CF set if data invalid
Note: this function is called by the DOS 3.1+ kernel
SeeAlso: INT 21/AH=36h
--------N-2F110D-----------------------------
INT 2F CU - IFSFUNC.EXE (DOS 4.x only) - SET REMOTE FILE'S ATTRIBUTES
  AX = 110Dh
  SDA first filename pointer -> name of file
  ???
Return: ???
Note: similar to AX=110Eh
SeeAlso: AX=110Eh
--------N-2F110E-----------------------------
INT 2F CU - NETWORK REDIRECTOR - SET REMOTE FILE'S ATTRIBUTES
  AX = 110Eh
  SS = DOS DS
  SDA first filename pointer -> fully-qualified name of file
  SDA CDS pointer -> current directory structure for drive with file
```

```
  STACK: WORD new file attributes
Return: CF set on error
      AX = DOS error code (see #01680 at INT 21/AH=59h/BX=0000h)
  CF clear if successful
  STACK unchanged
Note: this function is called by the DOS 3.1+ kernel
SeeAlso: AX=110Dh,AX=110Fh,INT 21/AX=4301h,INT 21/AH=60h
--------N-2F110F-----------------------------
INT 2F CU - NETWORK REDIRECTOR - GET REMOTE FILE'S ATTRIBUTES AND SIZE
  AX = 110Fh
  SS = DOS DS
  SDA first filename pointer -> fully-qualified name of file
  SDA CDS pointer -> current directory structure for drive with file
    (offset = FFFFh if null CDS [net direct request])
  SDA search attributes = mask of attributes which may be included in
      search for file
Return: CF set on error
      AX = DOS error code (see #01680 at INT 21/AH=59h/BX=0000h)
  CF clear if successful
      AX = file attributes
      BX:DI = file size
      CX = time stamp of file
      DX = date stamp of file
Notes:  this function is called by the DOS 3.1+ kernel
  wildcards and device names are not permitted
SeeAlso: AX=110Eh,INT 21/AX=4300h,INT 21/AH=60h
--------N-2F1110-----------------------------
INT 2F CU - IFSFUNC.EXE (DOS 4.x only) - GET REMOTE FILE'S ATTRIBUTES AND SIZE
  AX = 1110h
  SDA first filename pointer -> name of file
  ???
Return: ???
Note: appears to be similar to AX=110Fh
SeeAlso: AX=110Eh
--------N-2F1111-----------------------------
INT 2F CU - NETWORK REDIRECTOR - RENAME REMOTE FILE
  AX = 1111h
  SS = DS = DOS DS
  SDA first filename pointer = offset of fully-qualified old name
  SDA second filename pointer = offset of fully-qualified new name
  SDA CDS pointer -> current directory structure for drive with file
```

```
Return: CF set on error
      AX = DOS error code (see #01680 at INT 21/AH=59h/BX=0000h)
  CF clear if successful
Note: this function is called by the DOS 3.1+ kernel
SeeAlso: AX=1112h,INT 21/AH=56h,INT 21/AH=60h
--------N-2F1112-----------------------------
INT 2F CU - IFSFUNC.EXE (DOS 4.x only) - RENAME REMOTE FILE
  AX = 1112h
  SS = DS = DOS DS
  SDA first filename pointer -> name of file
  ???
Return: ???
Note: similar to AX=1111h
SeeAlso: AX=1111h
--------N-2F1113-----------------------------
INT 2F CU - NETWORK REDIRECTOR - DELETE REMOTE FILE
  AX = 1113h
  SS = DS = DOS DS
  SDA first filename pointer -> fully-qualified filename in DOS DS
  SDA CDS pointer -> current directory structure for drive with file
Return: CF set on error
      AX = DOS error code (see #01680 at INT 21/AH=59h/BX=0000h)
  CF clear if successful
Notes:  this function is called by the DOS 3.1+ kernel
  the filespec may contain wildcards
SeeAlso: AX=1114h,INT 21/AH=41h,INT 21/AH=60h
--------N-2F1114-----------------------------
INT 2F CU - IFSFUNC.EXE (DOS 4.x only) - DELETE REMOTE FILE
  AX = 1114h
  SDA first filename pointer -> name of file
  ???
Return: ???
Note: similar to AX=1113h
SeeAlso: AX=1113h
--------N-2F1115-----------------------------
INT 2F CU - IFSFUNC.EXE (DOS 4.x only) - OPEN REMOTE FILE
  AX = 1115h
  SS = DOS DS
  ES:DI -> SFT ???
  ???
Return: ???
```

```
Note: similar to AX=1116h
SeeAlso: AX=1116h,AX=112Eh
--------N-2F1116-----------------------------
INT 2F CU - NETWORK REDIRECTOR - OPEN EXISTING REMOTE FILE
  AX = 1116h
  ES:DI -> uninitialized SFT
  SS = DOS DS
  SDA first filename pointer -> fully-qualified name of file to open
  STACK: WORD file access and sharing modes (see #01402 at INT 21/AH=3Dh)
Return: CF set on error
      AX = DOS error code (see #01680 at INT 21/AH=59h/BX=0000h)
  CF clear if successful
      SFT filled (except handle count, which DOS manages itself)
  STACK unchanged
Note: this function is called by the DOS 3.1+ kernel
SeeAlso: AX=1106h,AX=1115h,AX=1117h,AX=1118h,AX=112Eh,INT 21/AH=3Dh
SeeAlso: INT 21/AH=60h
--------N-2F1117-----------------------------
INT 2F CU - NETWORK REDIRECTOR - CREATE/TRUNCATE REMOTE FILE
  AX = 1117h
  ES:DI -> uninitialized SFT
  SS = DOS DS
  SDA first filename pointer -> fully-qualified name of file to open
  SDA CDS pointer -> current directory structure for drive with file
  STACK: WORD file creation mode
      low byte = file attributes (see #01401 at INT 21/AH=3Ch)
      high byte = 00h normal create, 01h create new file
Return: CF set on error
      AX = DOS error code (see #01680 at INT 21/AH=59h/BX=0000h)
  CF clear if successful
      SFT filled (except handle count, which DOS manages itself)
  STACK unchanged
Note: this function is called by the DOS 3.1+ kernel
SeeAlso: AX=1106h,AX=1116h,AX=1118h,AX=112Eh,INT 21/AH=3Ch,INT 21/AH=60h
--------N-2F1118-----------------------------
INT 2F CU - NETWORK REDIRECTOR - CREATE/TRUNCATE FILE WITHOUT CDS
  AX = 1118h
  ES:DI -> uninitialized SFT
  SS = DOS DS
  SDA first filename pointer -> fully-qualified name of file
  STACK: WORD file creation mode
```

```
      low byte = file attributes
      high byte = 00h normal create, 01h create new file
Return: ???
  STACK unchanged
Note: this function is called by the DOS 3.1+ kernel when creating a file
    on a drive for which the SDA CDS pointer has offset FFFFh
SeeAlso: AX=1106h,AX=1116h,AX=1117h,AX=112Eh,INT 21/AH=60h
--------N-2F1119-----------------------------
INT 2F CU - NETWORK REDIRECTOR - FIND FIRST FILE WITHOUT CDS
  AX = 1119h
  SS = DS = DOS DS
  [DTA] = uninitialized 21-byte findfirst search data
        (see #01626 at INT 21/AH=4Eh)
  SDA first filename pointer -> fully-qualified search template
  SDA search attribute = attribute mask for search
Return: CF set on error
      AX = DOS error code (see #01680 at INT 21/AH=59h/BX=0000h)
  CF clear if successful
      [DTA] = updated findfirst search data
        (bit 7 of first byte must be set)
      [DTA+15h] = standard directory entry for file (see #01352)
Notes:  this function is called by the DOS 3.1+ kernel
  DOS 4.x IFSFUNC returns CF set, AX=0003h
SeeAlso: AX=111Ah,AX=111Bh,INT 21/AH=1Ah
--------N-2F111A-----------------------------
INT 2F CU - IFSFUNC.EXE (DOS 4.x only) - FIND NEXT FILE WITHOUT CDS
  AX = 111Ah
  ???
Return: CF set
      AX = error code (03h for DOS 4.01 IFSFUNC)
Note: use AX=111Ch for DOS 5+
SeeAlso: AX=1119h,AX=111Ch
--------N-2F111B-----------------------------
INT 2F CU - NETWORK REDIRECTOR - FINDFIRST
  AX = 111Bh
  SS = DS = DOS DS
  [DTA] = uninitialized 21-byte findfirst search data
        (see #01626 at INT 21/AH=4Eh)
  SDA first filename pointer -> fully-qualified search template
  SDA CDS pointer -> current directory structure for drive with file
  SDA search attribute = attribute mask for search
```

```
Return: CF set on error
      AX = DOS error code (see #01680 at INT 21/AH=59h/BX=0000h)
  CF clear if successful
      [DTA] = updated findfirst search data
        (bit 7 of first byte must be set)
      [DTA+15h] = standard directory entry for file (see #01352)
Note: this function is called by the DOS 3.1+ kernel
SeeAlso: AX=1119h,AX=111Ch,INT 21/AH=1Ah,INT 21/AH=4Eh,INT 21/AH=60h
--------N-2F111C-----------------------------
INT 2F CU - NETWORK REDIRECTOR - FINDNEXT
  AX = 111Ch
  SS = DS = DOS DS
  ES:DI -> CDS
  ES:DI -> DTA (MSDOS v5.0)
  [DTA] = 21-byte findfirst search data
        (see #01626 at INT 21/AH=4Eh)
Return: CF set on error
      AX = DOS error code (see #01680 at INT 21/AH=59h/BX=0000h)
  CF clear if successful
      [DTA] = updated findfirst search data
        (bit 7 of first byte must be set)
      [DTA+15h] = standard directory entry for file (see #01352)
Note: this function is called by the DOS 3.1+ kernel
SeeAlso: AX=1119h,AX=111Bh,INT 21/AH=1Ah,INT 21/AH=4Fh
--------N-2F111D-----------------------------
INT 2F CU - NETWORK REDIRECTOR - CLOSE ALL REMOTE FILES FOR PROCESS (ABORT)
  AX = 111Dh
  SS = DOS DS
  SDA PSP segment field = PSP of terminating process
Return: nothing
Notes:  used when a process is aborted; the process being terminated is
    indicated by the "sharing PSP" field in the SDA (offset 1Ah/1Ch)
  this function is called by the DOS 3.1+ kernel
  closes all FCBs opened by process
SeeAlso: INT 21/AX=5D04h
--------N-2F111E-----------------------------
INT 2F CU - NETWORK REDIRECTOR - DO REDIRECTION
  AX = 111Eh
  SS = DOS DS
  STACK: WORD function to execute
    5F00h  get redirection mode
```

```
      BL = type (03h printer, 04h disk)
      Return: BH = state (00h off, 01h on)
    5F01h  set redirection mode
      BL = type (03h printer, 04h disk)
      BH = state (00h off, 01h on)
    5F02h  get redirection list entry
      BX = redirection list index
      DS:SI -> 16-byte local device name buffer
      ES:DI -> 128-byte network name buffer
      Return: must set user's BX to device type and CX to
        stored parameter value, using AX=1218h to get
        stack frame address
    5F03h  redirect device
      BL = device type (see INT 21/AX=5F03h)
      CX = stored parameter value
      DS:SI -> ASCIZ source device name
      ES:DI -> destination ASCIZ network path + ASCIZ passwd
    5F04h  cancel redirection
      DS:SI -> ASCIZ device name or network path
    5F05h  get redirection list extended entry
      BX = redirection list index
      DS:SI -> buffer for ASCIZ source device name
      ES:DI -> buffer for destination ASCIZ network path
      Return: BH = status flag
        BL = type (03h printer, 04h disk)
        CX = stored parameter value
        BP = NETBIOS local session number
    5F06h  similar to 5F05h???
Return: CF set on error
      AX = error code (see #01680 at INT 21/AH=59h/BX=0000h)
  STACK unchanged
Notes:  this function is called by the DOS 3.1+ kernel on INT 21/AH=5Fh
    (including LAN Manager calls)
  the PC Network 1.00 redirector does not support function 5F06h
SeeAlso: INT 21/AX=5F00h,INT 21/AX=5F01h,INT 21/AX=5F02h,INT 21/AX=5F03h
SeeAlso: INT 21/AX=5F04h,INT 21/AX=5F05h,INT 21/AX=5F06h
--------N-2F111F-----------------------------
INT 2F CU - NETWORK REDIRECTOR - PRINTER SETUP
  AX = 111Fh
  STACK: WORD function
    5E02h  set printer setup
```

```
    5E03h  get printer setup
    5E04h  set printer mode
    5E05h  get printer mode
Return: CF set on error
      AX = error code (see #01680 at INT 21/AH=59h/BX=0000h)
  STACK unchanged
Note: this function is called by the DOS 3.1+ kernel
SeeAlso: INT 21/AX=5E02h,INT 21/AX=5E03h,INT 21/AX=5E04h,INT 21/AX=5E05h
--------N-2F1120-----------------------------
INT 2F CU - NETWORK REDIRECTOR - FLUSH ALL DISK BUFFERS
  AX = 1120h
  DS = DOS DS
  ???
Return: CF clear (successful)
Notes:  this function is called by the DOS 3.1+ kernel
  uses CDS array pointer and LASTDRIVE= entries in DOS list of lists
SeeAlso: INT 21/AH=0Dh,INT 21/AX=5D01h
--------N-2F1121-----------------------------
INT 2F CU - NETWORK REDIRECTOR - SEEK FROM END OF REMOTE FILE
  AX = 1121h
  CX:DX = offset (in bytes) from end
  ES:DI -> SFT
      SFT DPB field -> DPB of drive with file
  SS = DOS DS
Return: CF set on error
      AL = DOS error code (see #01680 at INT 21/AH=59h/BX=0000h)
  CF clear if successful
      DX:AX = new file position
Note: this function is called by the DOS 3.1+ kernel, but only when seeking
    from the end of a file opened with sharing modes set in such a
    manner that another process is able to change the size of the file
    while it is already open
SeeAlso: AX=1228h,INT 21/AH=42h
--------N-2F1122-----------------------------
INT 2F CU - NETWORK REDIRECTOR - PROCESS TERMINATION HOOK
  AX = 1122h
  SS = DOS DS
  DS = PSP of process about to terminate
Return: ???
Notes:  this function is called by the DOS 3.1+ kernel
  after calling this function, the kernel calls INT 2F/AX=111Dh
```

```
SeeAlso: AX=111Dh,INT 21/AH=4Ch,INT 60/DI=0601h
--------N-2F1123-----------------------------
INT 2F CU - NETWORK REDIRECTOR - QUALIFY REMOTE FILENAME
  AX = 1123h
  DS:SI -> ASCIZ filename to canonicalize
  ES:DI -> 128-byte buffer for qualified name
Return: CF set if not resolved
Notes:  called by MS-DOS 3.1+ kernel, but not called by DR DOS 5.0 unless the
    filename matches the name of a character device
  called first when DOS attempts to resolve a filename (unless inside an
    AX=5D00h server call); if this fails, DOS resolves the name locally
SeeAlso: AX=1221h,INT 21/AH=60h
--------N-2F1124-----------------------------
INT 2F CU - NETWORK REDIRECTOR - TURN OFF REMOTE PRINTER
  AX = 1124h
  ES:DI -> SFT
  SS = DOS DS
  ???
Return: CX = ???
Note: this function is called by the DOS 3.1+ kernel if AX=1126h
    returns CF set
SeeAlso: AX=1126h
--------N-2F1125-----------------------------
INT 2F CU - NETWORK REDIRECTOR - REDIRECTED PRINTER MODE
  AX = 1125h
  STACK: WORD subfunction
    5D07h get print stream state
      Return: DL = current state
    5D08h set print stream state
      DL = new state
    5D09h finish print job
Return: CF set on error
      AX = error code (see #01680 at INT 21/AH=59h/BX=0000h)
  STACK unchanged
Note: this function is called by the DOS 3.1+ kernel
SeeAlso: INT 21/AX=5D07h,INT 21/AX=5D08h,INT 21/AX=5D09h
--------N-2F1126-----------------------------
INT 2F CU - NETWORK REDIRECTOR - REMOTE PRINTER ECHO ON/OFF
  AX = 1126h
  ES:DI -> SFT for file handle 4???
  SS = DOS DS???
```

```
  ???
Return: CF set on error
Notes:  this function is called by the DOS 3.1+ kernel
  called when print echoing (^P, ^PrtSc) changes state and STDPRN has
    bit 11 of the device information word in the SFT set
SeeAlso: AX=1124h
--------N-2F1127-----------------------------
INT 2F CU - IFSFUNC.EXE (DOS 4.x only) - UNUSED
  AX = 1127h
Return: CF set
      AX = 0001h (invalid function) (see #01680 at INT 21/AH=59h/BX=0000h)
--------N-2F1127BX4E57----------------------
INT 2F - NetWare 4.0 - REMOTE FILE COPY
  AX = 1127h
  BX = 4E57h ('NW') (signature identifying this as a NetWare call)
  SI = source file handle
  DI = destination file handle
  DX:CX = number of bytes to copy, starting at current file position
Return: CF clear if successful
  CF set on error
      AX = error code (05h,06h,0Bh,11h,3Bh) (see #01680)
  DX:CX = number of bytes successfully copied (file position updated)
Notes:  this is the only call which may be made directly to the NetWare
    redirector from an application
  COMMAND.COM's COPY and DOS's XCOPY reportedly call INT 21/AX=1127h in
    order to speed up copies between files on the same network server;
    if error code 11h (not same device) is returned, the copy is
    performed in the usual manner.  However, no such calls appear to
    be present in MS-DOS 6.22.
  From the DR DOS "Panther" BETA COMMAND.COM (1992/06/22) up to some of
    the Novell DOS 7 COMMAND.COM updates (1994/09/12), the shell made
    calls to INT 2F/AX=11F0h to attempt "remote server COPYing". However,
    this was removed from later releases of the shell because it stopped
    Performance Technologies' PowerLAN 3.1 working. (A successor of
    the DR-DOS 7.03 COMMAND.COM may possibly reintroduce this remote
    copy feature.  Probably it would then try both INT 2F/AX=1127h and
    INT 2F/AX=11F0h.)
SeeAlso: INT 2F/AX=11F0h
--------N-2F1128-----------------------------
INT 2F CU - IFSFUNC.EXE (DOS 4.x only) - UNUSED
  AX = 1128h
```

```
Return: CF set
      AX = 0001h (invalid function) (see #01680 at INT 21/AH=59h/BX=0000h)
--------N-2F1129-----------------------------
INT 2F CU - IFSFUNC.EXE (DOS 4.x only) - UNUSED
  AX = 1129h
Return: CF set
      AX = 0001h (invalid function) (see #01680 at INT 21/AH=59h/BX=0000h)
--------N-2F112A-----------------------------
INT 2F CU - IFSFUNC.EXE (DOS 4.x only) - CLOSE ALL FILES FOR PROCESS
  AX = 112Ah
  DS = DOS DS
  ???
Return: ???
Note: does something to each IFS driver
--------N-2F112B-----------------------------
INT 2F CU - IFSFUNC.EXE (DOS 4.x only) - GENERIC IOCTL
  AX = 112Bh
  SS = DOS DS
  CX = function/category
  DS:DX -> parameter block
  STACK: WORD value of AX on entry to INT 21 (440Ch or 440Dh)
  ???
Return: CF set on error
      AX = DOS error code (see #01680 at INT 21/AH=59h/BX=0000h)
  CF clear if successful
Note: this function is called by the DOS 4.0 kernel
--------N-2F112C-----------------------------
INT 2F CU - NETWORK REDIRECTOR (DOS 4.0+) - "UPDATE_CB" - ???
  AX = 112Ch
  SS = DOS DS
  SDA current SFT pointer -> SFT for file
  ???
Return: CF set on error
Note: called by SHARE in DOS 5.0-6.0
--------N-2F112D-----------------------------
INT 2F CU - IFSFUNC.EXE (DOS 4.x only) - EXTENDED ATTRIBUTES
  AX = 112Dh
  BL = subfunction (value of AL on INT 21)
      02h get extended attributes
      03h get extended attribute properties
      04h set extended attributes
```

```
    Return: CF clear
      else ???
    Return: CX = ??? (00h or 02h for DOS 4.01)
  ES:DI -> SFT for file
  SS = DOS DS
Return: DS = DOS DS
Note: this function is called by the DOS 4.0 kernel on INT 21/AX=5702h,
    INT 21/AX=5703h, and INT 21/AX=5704h
SeeAlso: INT 21/AX=5702h,INT 21/AX=5703h,INT 21/AX=5704h,INT 21/AH=6Eh
--------N-2F112E-----------------------------
INT 2F CU - NETWORK REDIRECTOR (DOS 4.0+) - EXTENDED OPEN/CREATE FILE
  AX = 112Eh
  SS = DS = DOS DS
  ES:DI -> uninitialized SFT for file
  STACK: WORD file attribute for created/truncated file
      low byte = file attributes
      high byte = 00h normal create/open, 01h create new file
  SDA first filename pointer -> fully-qualified filename
  SDA extended file open action = action code
        (see #01770 at INT 21/AX=6C00h)
  SDA extended file open mode = open mode for file (see INT 21/AX=6C00h)
Return: CF set on error
      AX = error code
  CF clear if successful
      CX = result code
    01h file opened
    02h file created
    03h file replaced (truncated)
      SFT initialized (except handle count, which DOS manages itself)
Note: this function is called by the DOS 4.0+ kernel
BUG:  this function is not called correctly under some DOS versions
    (at least 5.0 and 6.2):
      the file attribute on the stack is not correct if the action
        code is 11h,
      the result code in CX is not passed back to the application.
SeeAlso: AX=1115h,AX=1116h,AX=1117h,INT 21/AX=6C00h
--------N-2F112F-----------------------------
INT 2F CU - IFSFUNC.EXE (DOS 4.x only) - IFS IOCTL
  AX = 112Fh
  SS = DOS DS
  STACK: WORD function in low byte
```

```
    00h ???
        DS:SI -> Current Directory Structure???
        CL = drive (1=A:)
    01h ???
        DS:SI -> ???
        CL = file handle???
    02h ???
        DS:SI -> Current Directory Structure???
        DI = ???
        CX = drive (1=A:)
  ???
Return: CF set on error
      AX = DOS error code (see #01680 at INT 21/AH=59h/BX=0000h)
  CF clear if successful
Note: this function is called by the DOS 4.0 kernel
SeeAlso: INT 21/AH=6Bh
--------N-2F1130-----------------------------
INT 2F CU - IFSFUNC.EXE (DOS 4.x only) - GET IFSFUNC SEGMENT
  AX = 1130h
Return: ES = CS of resident IFSFUNC
--------N-2F1180-----------------------------
INT 2F - LAN Manager Enhanced DOS Services - ???
  AX = 1180h
  ???
Return: ???
SeeAlso: AX=1100h,AX=1181h,AX=118Eh
--------N-2F1181-----------------------------
INT 2F - LAN Manager Enhanced DOS Services - SET USER NAME???
  AX = 1181h
  ???
Return: ???
SeeAlso: AX=1100h,AX=1180h
--------N-2F1182-----------------------------
INT 2F - LAN Manager Enhanced DOS Services - INSTALL SERVICE
  AX = 1182h
  ???
Return: ???
SeeAlso: AX=1100h,AX=1180h
--------N-2F1184-----------------------------
INT 2F - LAN Manager Enhanced DOS - ???
  AX = 1184h
```

```
  ???
Return: ???
--------N-2F1186-----------------------------
INT 2F - LAN Manager Enhanced DOS - DosReadAsynchNmPipe
  AX = 1186h
  DS:SI -> stack frame (see #02607)
Return: CF clear if successful
  CF set if error
      AX = error code
Note: LAN Manager enhance mode adds features beyond the standard redirector
    file/printer services
SeeAlso: AX=118Fh,AX=1190h,AX=1191h,INT 21/AX=5F39h

Format of LAN Manager DosReadAsynchNmPipe stack frame:
Offset  Size  Description (Table 02607)
 00h  DWORD -> number of bytes read
 04h  WORD  size of buffer
 06h  DWORD -> buffer
 0Ah  DWORD -> return code
 0Eh  DWORD function to call on completion as function( char far *buffer )
 12h  WORD  handle
--------N-2F118A-----------------------------
INT 2F - LAN Manager 2.0+ DOS Enhanced ENCRYPT.EXE - STREAM ENCRYPTION SERVICE
  AX = 118Ah
  BX = function (0000h or 0001h)
Return: CF clear if successful
      AX = 1100h success
  CF set if error
      AX = 0001h, etc.
SeeAlso: AX=1186h,AH=41h,AH=42h,AH=4Bh
--------N-2F118B-----------------------------
INT 2F - LAN Manager Enhanced DOS - ???
  AX = 118Bh
  ???
Return: ???
--------N-2F118C-----------------------------
INT 2F - LAN Manager Enhanced DOS - ???
  AX = 118Ch
  ???
Return: ???
--------N-2F118E-----------------------------
```

```
INT 2F - LAN Manager Enhanced DOS - ???
  AX = 118Eh
  ???
Return: ???
--------N-2F118F-----------------------------
INT 2F - LAN Manager Enhanced DOS - DosWriteAsynchNmPipe
  AX = 118Fh
  DS:SI -> stack frame (see #02608)
Return: CF clear if successful
  CF set if error
      AX = error code
SeeAlso: AX=1186h,AX=1191h,INT 21/AX=5F3Ah

Format of LAN Manager DosReadAsynchNmPipe stack frame:
Offset  Size  Description (Table 02608)
 00h  DWORD -> number of bytes read
 04h  WORD  Size of buffer
 06h  DWORD -> buffer
 0Ah  DWORD -> return code
 0Eh  DWORD function to call on completion as function( char far *buffer )
 12h  WORD  handle
--------N-2F1190-----------------------------
INT 2F - LAN Manager Enhanced DOS - DosReadAsynchNmPipe2
  AX = 1190h
  DS:SI -> stack frame (see #02609)
Return: CF clear if successful
  CF set if error
      AX = error code
SeeAlso: AX=1186h,AX=1191h

Format of LAN Manager DosReadAsynchNmPipe2 stack frame:
Offset  Size  Description (Table 02609)
 00h  DWORD -> number of bytes read
 04h  WORD  size of buffer
 06h  DWORD -> buffer
 0Ah  DWORD -> return code
 0Eh  DWORD function to call on completion as function( char far *buffer )
 12h  WORD  handle
 14h  DWORD ???
--------N-2F1191-----------------------------
INT 2F - LAN Manager Enhanced DOS - DosWriteAsynchNmPipe2
```

```
  AX = 1191h
  DS:SI -> stack frame (see #02610)
Return: CF clear if successful
  CF set if error
      AX = error code
SeeAlso: AX=118Fh,AX=1190h,INT 21/AX=5F3Ah

Format of LAN Manager DosReadAsynchNmPipe2 stack frame:
Offset  Size  Description (Table 02610)
 00h  DWORD -> number of bytes read
 04h  WORD  size of buffer
 06h  DWORD -> buffer
 0Ah  DWORD -> return code
 0Eh  DWORD function to call on completion as function( char far *buffer )
 12h  WORD  handle
 14h  DWORD ???
--------N-2F11F0-----------------------------
INT 2F - Novell ??? - REMOTE FILE COPY
  AX = 11F0h
  SI = source file handle
  DI = destination file handle
  DX:CX = number of bytes to copy, starting at current file position
  CF cleared
Return: CF clear:
      AX = 11F0h  "no network there"
      AX <> 11F0h if successful
  CF set on error "the request could not be handled"
      AX = error code (05h,06h,0Bh,11h,3Bh) (see #01680)
  DX:CX = number of bytes successfully copied (file position updated)
Notes:  From the DR DOS "Panther" BETA COMMAND.COM (1992/06/22) up to some of
    the Novell DOS 7 COMMAND.COM updates (1994/09/12), the shell made
    calls to INT 2F/AX=11F0h to attempt "remote server COPYing". However,
    this was removed from later releases of the shell because it
    interfered with Performance Technologies' PowerLAN 3.1. (A successor
    of the DR-DOS 7.03 COMMAND.COM may possibly reintroduce this remote
    copy feature. Probably it would then try both INT 2F/AX=1127h and
    INT 2F/AX=11F0h.)
SeeAlso: INT 2F/AX=1127h
--------D-2F1200-----------------------------
INT 2F U - DOS 3.0+ internal - INSTALLATION CHECK
  AX = 1200h
```

```
Return: AL = FFh (for compatibility with other INT 2F functions)
--------D-2F1201-----------------------------
INT 2F U - DOS 3.0+ internal - CLOSE CURRENT FILE
  AX = 1201h
  SS = DOS DS = DOS kernel data seg (must be using a DOS internal stack)
  SDA current SFT pointer -> SFT of file to close
Return: CF set on error
  ES:DI -> SFT for file
  CX undefined (new reference count of SFT in many versions)
  BX destroyed
SeeAlso: AX=1106h,AX=1227h,INT 21/AH=3Eh
--------D-2F1202-----------------------------
INT 2F U - DOS 3.0+ internal - GET INTERRUPT ADDRESS
  AX = 1202h
  STACK: WORD vector number
Return: ES:BX -> interrupt vector (DWORD containing handler's address)
  STACK unchanged
--------D-2F1203-----------------------------
INT 2F U - DOS 3.0+ internal - GET DOS DATA SEGMENT
  AX = 1203h
Return: DS = data segment of IBMDOS.COM/MSDOS.SYS
Note: for DOS prior to version 5.0, the data segment is the same as the code
    segment
--------D-2F1204-----------------------------
INT 2F U - DOS 3.0+ internal - NORMALIZE PATH SEPARATOR
  AX = 1204h
  STACK: WORD character to normalize
Return: AL = normalized character (forward slash turned to backslash, all
      others unchanged)
  ZF set if path separator
  STACK unchanged
--------D-2F1205-----------------------------
INT 2F U - DOS 3.0+ internal - OUTPUT CHARACTER TO STANDARD OUTPUT
  AX = 1205h
  STACK: WORD character to output
Return: STACK unchanged
Note: can be called only from within DOS
--------D-2F1206-----------------------------
INT 2F U - DOS 3.0+ internal - INVOKE CRITICAL ERROR
  AX = 1206h
  DI = error code
```

```
  BP:SI -> device driver header (see #01646)
  SS = DOS DS (must be using a DOS internal stack)
  STACK: WORD value to be passed to INT 24 in AX
Return: AL = 0-3 for Abort, Retry, Ignore, Fail
  STACK unchanged
SeeAlso: INT 24
--------D-2F1207-----------------------------
INT 2F U - DOS 3.0+ internal - MAKE DISK BUFFER MOST-RECENTLY USED
  AX = 1207h
  DS:DI -> disk buffer
Return: nothing
Desc: move the indicated buffer to the end of the disk buffer chain (least-
    recently used is first); under DOS 3.3, the buffer is then moved to
    the start of the disk buffer chain if it was marked unused
Notes:  can be called only from within DOS
  this function is nearly the same as AX=120Fh
SeeAlso: AX=120Fh
--------D-2F1208-----------------------------
INT 2F U - DOS 3.0+ internal - DECREMENT SFT REFERENCE COUNT
  AX = 1208h
  ES:DI -> SFT
Return: AX = original value of reference count
Notes:  if the reference count was 1, it is set to FFFFh ("busy", since 0
    indicates that the SFT is not in use).  It is the caller's
    responsibility to set the reference count to zero after cleaning up.
  used by network redirectors such as MSCDEX
SeeAlso: AX=1106h
--------D-2F1209-----------------------------
INT 2F U - DOS 3.0+ internal - FLUSH AND FREE DISK BUFFER
  AX = 1209h
  DS:DI -> disk buffer
Return: disk buffer marked unused, contents written to disk if buffer dirty
Note: can be called only from within DOS
SeeAlso: AX=120Eh,AX=1215h
--------D-2F120A-----------------------------
INT 2F U - DOS 3.0+ internal - PERFORM CRITICAL ERROR INTERRUPT
  AX = 120Ah
  DS = SS = DOS DS (must be using a DOS internal stack)
  STACK: WORD extended error code
Return: AL = user response (0=ignore, 1=retry, 2=abort, 3=fail)
  CF clear if retry, set otherwise
```

```
  STACK unchanged
Notes:  can only be called during a DOS function call, as it uses various
    fields in the SDA to set up the registers for the INT 24
  reportedly sets current DPB's first root directory sector to 1
SeeAlso: INT 24
--------D-2F120B-----------------------------
INT 2F U - DOS 3.0+ internal - SIGNAL SHARING VIOLATION TO USER
  AX = 120Bh
  ES:DI -> system file table entry for previous open of file
  STACK: WORD extended error code (should be 20h--sharing violation)
Return: CF clear if operation should be retried
  CF set if operation should not be retried
      AX = error code (20h) (see #01680 at INT 21/AH=59h/BX=0000h)
  STACK unchanged
Notes:  can only be called during a DOS function call
  should only be called if an attempt was made to open an already-open
    file contrary to the sharing rules
  invokes INT 24 if SFT file opened via FCB or in compatibility mode with
    inheritance allowed
--------D-2F120C-----------------------------
INT 2F U - DOS 3.0+ internal - OPEN DEVICE AND SET SFT OWNER/MODE
  AX = 120Ch
  SDA current SFT pointer -> SFT for file
  DS = DOS DS
  SS = DOS DS (must be using a DOS internal stack)
Return: ES, DI, AX destroyed
Notes:  invokes "device open" call on device driver for SFT
  changes owner of last-accessed SFT to calling process if it was opened
    via FCB
  called by network redirectors such as MSCDEX
--------D-2F120D-----------------------------
INT 2F U - DOS 3.0+ internal - GET DATE AND TIME
  AX = 120Dh
  SS = DOS DS (must be using a DOS internal stack)
Return: AX = current date in packed format (see #01666 at INT 21/AX=5700h)
  DX = current time in packed format (see #01665 at INT 21/AX=5700h)
SeeAlso: INT 21/AH=2Ah,INT 21/AH=2Ch
--------D-2F120E-----------------------------
INT 2F U - DOS 3.0+ internal - MARK ALL DISK BUFFERS UNREFERENCED
  AX = 120Eh
  SS = DOS DS (must be using a DOS internal stack)
```

```
Return: DS:DI -> first disk buffer
Notes:  clears "referenced" flag on all disk buffers
  in DOS 5+, this has become essentially a NOP, invoking the same code
    used by AX=1224h (SHARING DELAY)
SeeAlso: AX=1209h,AX=1210h,INT 21/AH=0Dh
--------D-2F120F-----------------------------
INT 2F U - DOS 3.0+ internal - MAKE BUFFER MOST RECENTLY USED
  AX = 120Fh
  DS:DI -> disk buffer
  SS = DOS DS (must be using a DOS internal stack)
Return: DS:DI -> next buffer in buffer list
Desc: move the indicated buffer to the end of the disk buffer chain (least-
    recently used is first); under DOS 3.3, the buffer is then moved to
    the start of the disk buffer chain if it was marked unused
Note: this function is the same as AX=1207h except that it returns a
    pointer to the buffer following the specified buffer in the buffer
    chain
SeeAlso: AX=1207h
--------D-2F1210-----------------------------
INT 2F U - DOS 3.0+ internal - FIND UNREFERENCED DISK BUFFER
  AX = 1210h
  DS:DI -> first disk buffer to check
Return: ZF clear if found
      DS:DI -> first unreferenced disk buffer
  ZF set if not found
Note: in DOS 5+, this has become essentially a NOP, invoking the same code
    used by AX=1224h (SHARING DELAY)
SeeAlso: AX=120Eh
--------D-2F1211-----------------------------
INT 2F U - DOS 3.0+ internal - NORMALIZE ASCIZ FILENAME
  AX = 1211h
  DS:SI -> ASCIZ filename to normalize
  ES:DI -> buffer for normalized filename
Return: destination buffer filled with uppercase filename, with slashes turned
  to backslashes
SeeAlso: AX=121Eh,AX=1221h
--------D-2F1212-----------------------------
INT 2F U - DOS 3.0+ internal - GET LENGTH OF ASCIZ STRING
  AX = 1212h
  ES:DI -> ASCIZ string
Return: CX = length of string
```

```
SeeAlso: AX=1225h
--------D-2F1213-----------------------------
INT 2F U - DOS 3.0+ internal - UPPERCASE CHARACTER
  AX = 1213h
  STACK: WORD character to convert to uppercase
Return: AL = uppercase character
  STACK unchanged
--------D-2F1214-----------------------------
INT 2F U - DOS 3.0+ internal - COMPARE FAR POINTERS
  AX = 1214h
  DS:SI = first pointer
  ES:DI = second pointer
Return: ZF set if pointers are equal, ZF clear if not equal
  CF clear if pointers equal, CF set if not
--------D-2F1215-----------------------------
INT 2F U - DOS 3.0+ internal - FLUSH BUFFER
  AX = 1215h
  DS:DI -> disk buffer
  SS = DOS DS (must be using a DOS internal stack)
  STACK: WORD drives for which to skip buffer
    ignore buffer if drive same as high byte, or bytes differ and
      the buffer is for a drive OTHER than that given in low byte
Return: STACK unchanged
Note: can be called only from within DOS
SeeAlso: AX=1209h
--------D-2F1216-----------------------------
INT 2F U - DOS 3.0+ internal - GET ADDRESS OF SYSTEM FILE TABLE ENTRY
  AX = 1216h
  BX = system file table entry number
Return: CF clear if successful
      ES:DI -> system file table entry
      BX = relative entry number in system file table containing entry
      AX destroyed
  CF set if BX greater than FILES=
Note: supported by DR DOS 5+
SeeAlso: AX=1220h
--------D-2F1217-----------------------------
INT 2F U - DOS 3.0+ internal - GET CURRENT DIRECTORY STRUCTURE FOR DRIVE
  AX = 1217h
  SS = DOS DS (must be using a DOS internal stack)
  STACK: WORD drive (0 = A:, 1 = B:, etc)
```

```
Return: CF set on error
      (drive > LASTDRIVE)
  CF clear if successful
      DS:SI -> current directory structure for specified drive
  STACK unchanged
SeeAlso: AX=1219h
--------D-2F1218-----------------------------
INT 2F U - DOS 3.0+ internal - GET CALLER'S REGISTERS
  AX = 1218h
Return: DS:SI -> saved caller's AX,BX,CX,DX,SI,DI,BP,DS,ES (on stack)
Note: only valid while within DOS
--------D-2F1219-----------------------------
INT 2F U - DOS 3.0+ internal - SET DRIVE???
  AX = 1219h
  SS = DOS DS (must be using a DOS internal stack)
  STACK: WORD drive (0 = default, 1 = A:, etc)
Return: ???
  STACK unchanged
Notes:  calls AX=1217h
  builds a current directory structure if inside server call
    (INT 21/AX=5D00h)
SeeAlso: AX=1217h,AX=121Fh
--------D-2F121A-----------------------------
INT 2F U - DOS 3.0+ internal - GET FILE'S DRIVE
  AX = 121Ah
  DS:SI -> filename
Return: AL = drive (0 = default, 1 = A:, etc, FFh = invalid)
  DS:SI -> filename without leading X: (if present)
SeeAlso: INT 21/AH=19h,INT 21/AH=60h
--------D-2F121B-----------------------------
INT 2F U - DOS 3.0+ internal - SET YEAR/LENGTH OF FEBRUARY
  AX = 121Bh
  CL = year - 1980
Return: AL = number of days in February
Note: requires DS to be set to the DOS data segment
SeeAlso: INT 21/AH=2Bh"DATE"
--------D-2F121C-----------------------------
INT 2F U - DOS 3.0+ internal - CHECKSUM MEMORY
  AX = 121Ch
  DS:SI -> start of memory to checksum
  CX = number of bytes
```

```
  DX = initial checksum
  SS = DOS DS (must be using a DOS internal stack)
Return: AX, CX destroyed
  DX = checksum
  DS:SI -> first byte after checksummed range
Notes:  used by DOS to determine day count since 1980/1/1 given a date
  supported by DR DOS 5.0+
SeeAlso: AX=121Dh
--------D-2F121D-----------------------------
INT 2F U - DOS 3.0+ internal - SUM MEMORY
  AX = 121Dh
  DS:SI -> memory to add up
  CX = 0000h
  DX = limit
Return: AL = byte which exceeded limit
  CX = number of bytes before limit exceeded
  DX = remainder after adding first CX bytes
  DS:SI -> byte beyond the one which exceeded the limit
Notes:  used by DOS to determine year or month given day count since 1980/1/1
  supported by DR DOS 5.0+
SeeAlso: AX=121Ch
--------D-2F121E-----------------------------
INT 2F U - DOS 3.0+ internal - COMPARE FILENAMES
  AX = 121Eh
  DS:SI -> first ASCIZ filename
  ES:DI -> second ASCIZ filename
Return: ZF set if filenames equivalent, ZF clear if not
Note: supported by DR DOS 5.0+
SeeAlso: AX=1211h,AX=1221h
--------D-2F121F-----------------------------
INT 2F U - DOS 3.0+ internal - BUILD CURRENT DIRECTORY STRUCTURE
  AX = 121Fh
  SS = DOS DS (must be using a DOS internal stack)
  STACK: WORD drive letter
Return: ES:DI -> current directory structure (will be overwritten by next call)
  STACK unchanged
--------D-2F1220-----------------------------
INT 2F U - DOS 3.0+ internal - GET JOB FILE TABLE ENTRY
  AX = 1220h
  BX = file handle
Return: CF set on error
```

```
      AL = 6 (invalid file handle)
  CF clear if successful
      ES:DI -> JFT entry for file handle in current process
Notes:  the byte pointed at by ES:DI contains the number of the SFT for the
    file handle, or FFh if the handle is not open
  supported by DR DOS 5.0+
SeeAlso: AX=1216h,AX=1229h
--------D-2F1221----------------------------
INT 2F U - DOS 3.0+ internal - CANONICALIZE FILE NAME
  AX = 1221h
  DS:SI -> file name to be fully qualified
  ES:DI -> 128-byte buffer for resulting canonical file name
  SS = DOS DS (must be using a DOS internal stack)
Return: (see INT 21/AH=60h)
Note: identical to INT 21/AH=60h
SeeAlso: AX=1123h,INT 21/AH=60h
--------D-2F1222----------------------------
INT 2F U - DOS 3.0+ internal - SET EXTENDED ERROR INFO
  AX = 1222h
  SS = DOS data segment
  SS:SI -> 4-byte records
    BYTE  error code, FFh = last record
    BYTE  error class, FFh = don't change
    BYTE  suggested action, FFh = don't change
    BYTE  error locus, FFh = don't change
  SDA error code set
Return: SI destroyed
  SDA error class, error locus, and suggested action fields set
Note: can be called only from within DOS
SeeAlso: AX=122Dh,INT 21/AH=59h/BX=0000h,INT 21/AX=5D0Ah
--------D-2F1223----------------------------
INT 2F U - DOS 3.0+ internal - CHECK IF CHARACTER DEVICE
  AX = 1223h
  DS = DOS DS
  SS = DOS DS (must be using a DOS internal stack)
  SDA+218h (DOS 3.10-3.30) = eight-character blank-padded name
  SDA+22Bh (DOS 4.0-6.0) = eight-character blank-padded name
  SDA file attribute field set
  direction flag clear (i.e. CLD)
Return: CF set if no character device by that name found
  CF clear if found
```

```
      BH bits 4-0 copied from low byte of device attribute word
      BH bit 5 set, bits 7-6 clear
Notes:  can only be called from within DOS
  the check is skipped (always says "not device") if the volume ID bit
    of the file attribute field is set on entry
SeeAlso: INT 21/AX=5D06h,INT 21/AX=5D0Bh
--------D-2F1224-----------------------------
INT 2F U - DOS 3.0+ internal - SHARING RETRY DELAY
  AX = 1224h
  SS = DOS DS (must be using a DOS internal stack)
Return: after delay set by INT 21/AX=440Bh, unless in server call
    (INT 21/AX=5D00h)
Note: delay is dependent on the processor speed, and is skipped entirely if
    inside a server call
SeeAlso: INT 21/AX=440Bh,INT 21/AH=52h,INT 62/AX=0097h
--------D-2F1225-----------------------------
INT 2F U - DOS 3.0+ internal - GET LENGTH OF ASCIZ STRING
  AX = 1225h
  DS:SI -> ASCIZ string
Return: CX = length of string
Note: supported by DR DOS 5.0+
SeeAlso: AX=1212h
--------D-2F1226-----------------------------
INT 2F U - DOS 3.3+ internal - OPEN FILE
  AX = 1226h
  CL = access mode
  DS:DX -> ASCIZ filename
  SS = DOS DS (must be using a DOS internal stack)
Return: CF set on error
      AL = error code (see #01680 at INT 21/AH=59h/BX=0000h)
  CF clear if successful
      AX = file handle
Notes:  can only be called from within DOS
  equivalent to INT 21/AH=3Dh
  used by NLSFUNC to access COUNTRY.SYS when invoked by the DOS kernel
SeeAlso: AX=1227h,INT 21/AH=3Dh
--------D-2F1227-----------------------------
INT 2F U - DOS 3.3+ internal - CLOSE FILE
  AX = 1227h
  BX = file handle
  SS = DOS DS (must be using a DOS internal stack)
```

```
Return: CF set on error
      AL = 06h invalid file handle
  CF clear if successful
Notes:  can only be called from within DOS
  equivalent to INT 21/AH=3Eh
  used by NLSFUNC to access COUNTRY.SYS when invoked by the DOS kernel
SeeAlso: AX=1106h,AX=1201h,AX=1226h,INT 21/AH=3Eh
--------D-2F1228BP4200----------------------
INT 2F U - DOS 3.3+ internal - MOVE FILE POINTER
  AX = 1228h
  BP = 4200h, 4201h, 4202h (see INT 21/AH=42h)
  BX = file handle
  CX:DX = offset in bytes
  SS = DOS DS (must be using a DOS internal stack)
Return: as for INT 21/AH=42h
Notes:  equivalent to INT 21/AH=42h, but may only be called from inside a DOS
    function call
  sets user stack frame pointer to dummy buffer, moves BP to AX, performs
    LSEEK, and restores frame pointer
  used by NLSFUNC to access COUNTRY.SYS when invoked by the DOS kernel
SeeAlso: INT 21/AH=42h
--------D-2F1229----------------------------
INT 2F U - DOS 3.3+ internal - READ FROM FILE
  AX = 1229h
  BX = file handle
  CX = number of bytes to read
  DS:DX -> buffer
  SS = DOS DS (must be using a DOS internal stack)
Return: as for INT 21/AH=3Fh"DOS"
Notes:  equivalent to INT 21/AH=3Fh, but may only be called when already inside
    a DOS function call
  used by NLSFUNC to access COUNTRY.SYS when invoked by the DOS kernel
SeeAlso: AX=1226h,INT 21/AH=3Fh"DOS"
--------D-2F122A----------------------------
INT 2F U - DOS 3.3+ internal - SET FASTOPEN ENTRY POINT
  AX = 122Ah
  BX = entry point to set (0001h or 0002h)
  DS:SI -> FASTOPEN entry point (see #02611,#02612)
    (entry point not set if SI = FFFFh for DOS 4.0+)
Return: CF set if specified entry point already set
Notes:  entry point in BX is ignored under DOS 3.30
```

```
  both entry points set to same handler by DOS 4.01
  DOS 5.0 and 6.0 only set entry point 1

(Table 02611)
Values DOS 3.30+ FASTOPEN entry point is called with:
  AL = 01h  Lookup
       CX = ??? seems to be offset
       DI = ??? seems to be offset
       SI = offset in DOS DS of filename
  AL = 02h  insert file into FASTOPEN cache
  AL = 03h  delete file from FASTOPEN cache
       SI = offset in DOS DS of filename
  AL = 04h  purge FASTOPEN cache
       AH = subfunction (00h,01h,02h)
       ES:DI -> ???
       CX = ??? (subfunctions 01h and 02h only)
Returns: CF set on error or not installed
Note: function 03h calls function 01h first
SeeAlso: #02612,#02613

(Table 02612)
Values PC-DOS 4.01 FASTOPEN is additionally called with:
  AL = 04h ???
       AH = 03h
       ???
  AL = 05h ???
  AL = 0Bh ???
  AL = 0Ch ???
  AL = 0Dh ???
  AL = 0Eh ???
  AL = 0Fh ???
  AL = 10h ???
SeeAlso: #02611,#02613

(Table 02613)
Values MS-DOS 5.0-6.0 FASTOPEN is additionally called with:
  AL = 04h  purge FASTOPEN cache
       AH = 03h
       ???
  AL = 05h ???
       DL = drive (00h = A:)
```

```
      ???
  AL = 06h ???
      ???
SeeAlso: #02611,#02612
--------D-2F122B-----------------------------
INT 2F U - DOS 3.3+ internal - IOCTL
  AX = 122Bh
  BP = 44xxh
  SS = DOS DS (must be using a DOS internal stack)
  additional registers as appropriate for INT 21/AX=44xxh
Return: as for INT 21/AH=44h
Notes:  equivalent to INT 21/AH=44h, but may only be called when already inside
    a DOS function call
  sets user stack frame pointer to dummy buffer, moves BP to AX, performs
    IOCTL, and restores frame pointer
  used by NLSFUNC in accessing COUNTRY.SYS when invoked by the DOS kernel
SeeAlso: INT 21/AH=44h
--------D-2F122C-----------------------------
INT 2F U - DOS 3.3+ internal - GET DEVICE CHAIN
  AX = 122Ch
Return: BX:AX -> header of second device driver (NUL is first) in driver chain
Note: although this function exists in DR DOS 5.0 and Novell DOS 7, it
    always returns 0000h:0000h prior to Novell DOS 7 Update 15
SeeAlso: INT 21/AH=52h
--------D-2F122D-----------------------------
INT 2F U - DOS 3.3+ internal - GET EXTENDED ERROR CODE
  AX = 122Dh
  SS = DOS DS
Return: AX = current extended error code
SeeAlso: AX=1222h,INT 21/AH=59h/BX=0000h
--------D-2F122E-----------------------------
INT 2F U - DOS 4.0+ internal - GET OR SET ERROR TABLE ADDRESSES
  AX = 122Eh
  DL = subfunction
      00h get standard DOS error table (see #02614)
    Return: ES:DI -> error table
          (DOS 4: errors 00h-12h,50h-5Bh)
          (DOS 5: errors 00h-26h,4Fh,51h-59h)
      01h set standard DOS error table
    ES:DI -> error table
      02h get parameter error table (errors 00h-0Ah)
```

```
    Return: ES:DI -> error table
      03h set parameter error table
    ES:DI -> error table
      04h get critical/SHARE error table (errors 13h-2Bh)
    Return: ES:DI -> error table
      05h set critical/SHARE error table
    ES:DI -> error table
      06h get ??? error table
    Return: ES:DI -> error table or 0000h:0000h
      07h set ??? error table
    ES:DI -> error table
      08h get error message retriever (see #02615)
    Return: ES:DI -> FAR procedure to fetch error message
      09h set ??? error table
    ES:DI -> error table
Notes:  if the returned segment on a "get" is 0001h, then the offset specifies
    the offset of the error message table within COMMAND.COM, and the
    procedure returned by DL=08h should be called
  DOS 5+ COMMAND.COM does not allow setting any of the addresses (calls
    with DL odd are ignored); they are always returned with segment 0001h
  for DOS 5.0, the standard and critical/SHARE error tables are combined
    into a single error table
SeeAlso: AX=0500h,INT 21/AH=59h/BX=0000h

Format of DOS 4.x error table:
Offset  Size  Description (Table 02614)
 00h  BYTE  FFh
 01h  2 BYTEs 04h,00h (DOS version???)
 03h  BYTE  number of error headers following
 04h 2N WORDs table of all error headers for table
    Offset  Size  Description
     00h  WORD  error message number
     02h  WORD  offset of error message from start of header
        error messages are count byte followed by msg
Note: DOS 5 error tables consist of one word per error number; each word
    contains either the offset of a counted string or 0000h

(Table 02615)
Call error retrieval function with:
  AX = error number (see #02616)
  DI = offset of error table
```

```
Return: ES:DI -> error message (counted string)
Notes:  this function needs to access COMMAND.COM if the messages were not
    loaded into memory permanently with /MSG; the caller should assume
    that the returned message will be overwritten by the next call of
    the function
  supported by DR DOS 5.0

(Table 02616)
Values for parameter errors:
 01h  Too many parameters
 02h  Required Parameter missing
 03h  Invalid switch
 04h  Invalid keyword
 06h  Parameter value not in allowed range
 07h  Parameter value not allowed
 08h  Parameter value not allowed
 09h  Parameter format not correct
 0Ah  Invalid parameter
 0Bh  Invalid parameter combination
--------D-2F122F-----------------------------
INT 2F U - DOS 4.x internal - SET DOS VERSION NUMBER TO RETURN
  AX = 122Fh
  DX = DOS version number (0000h = return true DOS version)
Notes:  not available under DR DOS 5.0 or 6.0, or Novell DOS 7
  supported by FREEVER.COM, a freeware DOS version faking TSR by Matthias
    Paul
SeeAlso: INT 21/AH=30h,INT 21/AX=3306h,INT 2F/AX=E000h"SETDRVER"
----------2F1230-----------------------------
INT 2F U - Windows95 - FIND SFT ENTRY IN INTERNAL FILE TABLES
  AX = 1230h
  ES:DI -> SFT entry
Return: CF clear if SFT found in internal table
  CF set if SFT not in any internal file table
      AX = 0000h
      SI:CX = 32-bit starting cluster number for directory
      DX = directory entry number
      BX = index into new file system table
Notes:  the new file system table from which the return values are taken is
    reported to be statically allocated with 20 entries, and used only
    for FCB calls
  this function is not supported by DR-DOS 7.03 or earlier, by S/DOS 1.0,
```

```
    or by PTS-DOS 6.51
BUG:  Win95-OSR2 is reported to have a bug that can potentially corrupt
    memory if SFT tables are "arranged poorly"
SeeAlso: AX=1231h,AX=1200h
----------2F1231-----------------------------
INT 2F U - Windows95 - SET/CLEAR "REPORT WINDOWS TO DOS PROGRAMS" FLAG
  AX = 1231h
  DL = function
       00h set byte after "IsWIN386" to 01h
       01h set "IsWIN386" bit 1
       02h clear "IsWIN386" bit 1
       else
    Return: CF set
       AX = 0001h
Return: CF clear
  AX = 0000h
Note: this function is not supported by DR-DOS 7.03 or earlier, by S/DOS 1.0,
    or by PTS-DOS 6.51
BUG:  Windows98 will crash the system if DL>02h on entry due to an off-by-1
    conditional jump; if the jump were correct, the function would return
    CF set/AX=0001h as for Windows95
SeeAlso: AX=1230h,AX=1200h
--------O-2F1250-----------------------------
INT 2F U - PTS-DOS - SET MACHINE ID
  AX = 1250h
  ???
Return: ???
Note: This is known to be supported by Paragon Technology Systems PTS-DOS
    sometime before 6.51, but is known not to be supported by S/DOS 1.0
    (which derived from PTS-DOS 6.51)
SeeAlso: AX=1251h
--------O-2F1251-----------------------------
INT 2F U - PTS-DOS - GET MACHINE ID
  AX = 1251h
  ???
Return: ???
Note: This is known to be supported by Paragon Technology Systems PTS-DOS
    sometime before 6.51, but is known not to be supported by S/DOS 1.0
    (which derived from PTS-DOS 6.51)
SeeAlso: AX=1250h
--------O-2F1252-----------------------------
```

```
INT 2F U - PTS-DOS 6.51, S/DOS 1.0+ - SET SFT INCREMENT
  AX = 1252h
  BX = new SFT increment
Return: ???
Note: This function is known to be supported by Paragon Technology Systems
    PTS-DOS 6.51 and S/DOS 1.0 and probably was also supported
    in earlier releases.
SeeAlso: AX=1253h
--------O-2F1253-----------------------------
INT 2F U - PTS-DOS 6.51, S/DOS 1.0+ - GET SFT INCREMENT
  AX = 1253h
Return: BX = current SFT increment
  AX = ???
Note: This function is known to be supported by Paragon Technology Systems
    PTS-DOS 6.51 and S/DOS 1.0 and probably was also supported
    in earlier releases.
SeeAlso: AX=1252h
--------O-2F1260-----------------------------
INT 2F C - PTS-DOS 6.51, S/DOS 1.0+ - "EMPTY" (RESERVED FOR INPUT STRING)
  AX = 1260h
  ES:DI -> string buffer (see getstr)
  DL = extended keystroke code or FFh if ENTER
Return: AX = 0000h (default handler in the kernel)
  CF set if action done, all registers preserved
Notes:  This function is known to be handled by Paragon Technology Systems
    PTS-DOS 6.51 and S/DOS 1.0, and probably was also handled
    in earlier releases.
  The default handler in the kernel justs XORs AX,AX and returns.
  This is reserved to be invoked in input string
--------O-2F1261-----------------------------
INT 2F - PTS-DOS 6.51, S/DOS 1.0+ - GET FIRST UMB
  AX = 1261h
Return: AX = address of first UMB or 1 if invalid function.
Note: This function is known to be supported by Paragon Technology Systems
    PTS-DOS 6.51 and S/DOS 1.0, and probably was also supported
    in earlier releases.
--------O-2F1262-----------------------------
INT 2F - PTS-DOS 6.51, S/DOS 1.0+ - GET DOS COLOR
  AX = 1262h
Return: AL = current video attribute used by DOS functions
Note: This function is known to be supported by Paragon Technology Systems
```

```
    PTS-DOS 6.51 and S/DOS 1.0, and probably was also supported
    in earlier releases.
SeeAlso: AX=1263h
--------O-2F1263-----------------------------
INT 2F - PTS-DOS 6.51, S/DOS 1.0+ - SET DOS COLOR
  AX = 1263h
  DL = new video attribute to be used by DOS functions
Return: ???
Note: This function is known to be supported by Paragon Technology Systems
    PTS-DOS 6.51 and S/DOS 1.0, and probably was also supported
    in earlier releases.
SeeAlso: AX=1262h
--------O-2F1270-----------------------------
INT 2F - PTS-DOS 6.51, S/DOS 1.0+ - "SYSBELL" - EMIT A BEEP
  AX = 1270h
Return: ???
Note: This function is known to be supported by Paragon Technology Systems
    PTS-DOS 6.51 and S/DOS 1.0, and probably was also supported
    in earlier releases.
SeeAlso: AX=1271h
--------O-2F1271-----------------------------
INT 2F C - PTS-DOS, S/DOS - RESERVED FOR BEEP
  AX = 1271h
Return: ???
Note: According to the Paragon Technology Systems S/DOS 1.0 sources,
    which derived from PTS-DOS 6.51, this is reserved for a "BEEP"
    function. However, S/DOS 1.0 does not handle this function by
    itself.
SeeAlso: AX=1270h
--------O-2F12F0-----------------------------
INT 2F - PTS-DOS 6.51, S/DOS 1.0+ - BACKDOOR INTERCEPT INT 2Fh CHAIN
  AX = 12F0h
  CX:DX -> user INT 2F handler
  CF set
Return: CF clear if successful
      CX:DX -> previous INT 2F handler
  CF set on error
Notes:  This function is known to be supported by Paragon Technology Systems
    PTS-DOS 6.51 and S/DOS 1.0, and probably was also supported
    in earlier releases.  It is handled from within the kernel's
    INT 2Fh dispatcher.
```

```
  Resident system extensions should call INT 2Fh/12F0h to intercept
    the INT 2Fh chain. The main idea of this call is to allow them to
    be moved to the HMA and intercept INT 2Fh without querying A20 state
    and without interception of the INT2Fh vector. This is why the
    "Chain2F" variable must be instanced by placing it to the SDA.
  On chain entry of the far procedure, CF must be set.  All registers
    except for AX, BP, and DS remain unchanged. If a handler cannot
    process the call, it must set CF and do a far jump to the previous
    one in the chain in order to indicate an error.  If CF is still set
    on exit, the call will be ignored.
--------O-2F12F1-----------------------------
INT 2F - PTS-DOS, S/DOS - RESERVED
  AX = 12F1h-12FFh
Return: ???
Note: According to the Paragon Technology Systems S/DOS 1.0 sources,
    which derived from PTS-DOS 6.51, these functions are reserved for
    future use. However, S/DOS 1.0 does not make use of them itself.
--------O-2F12FFBL00------------------------
INT 2F - FreeDOS - FDAK-DDT - INSTALLATION CHECK / STATUS CHECK
  AX = 12FFh
  BL = 00h
Return: AL = DDh if installed
      BH = state (00h disabled, nonzero enabled)
      BL = readonly flag (00h writable, nonzero read-only)
Program: FDAK-DDT is the FreeDOS Alternative Kernel Device Drivers Testing
    release by Yury A. Semenov
SeeAlso: AX=12FFh/BL=07h
--------O-2F12FFBL01------------------------
INT 2F - FreeDOS - FDAK-DDT - ENABLE FDAK DRIVERS
  AX = 12FFh
  BL = 01h
SeeAlso: AX=12FFh/BL=00h,AX=12FFh/BL=02h
--------O-2F12FFBL02------------------------
INT 2F - FreeDOS - FDAK-DDT - DISABLE FDAK DRIVERS
  AX = 12FFh
  BL = 02h
SeeAlso: AX=12FFh/BL=00h,AX=12FFh/BL=01h
--------O-2F12FFBL03------------------------
INT 2F - FreeDOS - FDAK-DDT - SWITCH BLOCK DEVICE TO READ-ONLY
  AX = 12FFh
  BL = 03h
```

```
  ???
SeeAlso: AX=12FFh/BL=00h,AX=12FFh/BL=04h
--------O-2F12FFBL04-------------------------
INT 2F - FreeDOS - FDAK-DDT - SWITCH BLOCK DEVICE TO READ-WRITE
  AX = 12FFh
  BL = 04h
  ???
SeeAlso: AX=12FFh/BL=00h,AX=12FFh/BL=03h
--------O-2F12FFBL05-------------------------
INT 2F - FreeDOS - FDAK-DDT - TURN ACTIVITY INDICATOR ON
  AX = 12FFh
  BL = 05h
Note: not yet implemented as of January 1996
SeeAlso: AX=12FFh/BL=00h,AX=12FFh/BL=06h
--------O-2F12FFBL06-------------------------
INT 2F - FreeDOS - FDAK-DDT - TURN ACTIVITY INDICATOR OFF
  AX = 12FFh
  BL = 06h
Note: not yet implemented as of January 1996
SeeAlso: AX=12FFh/BL=00h,AX=12FFh/BL=05h
--------O-2F12FFBL07-------------------------
INT 2F - FreeDOS - FDAK-DDT - UNINSTALL
  AX = 12FFh
  BL = 07h
Return: ES = segment of FDAK memory block
Note: the caller must free the memory block returned in ES
    (via INT 21/AH=49h)
SeeAlso: AX=12FFh/BL=00h
--------O-2F12FFBX0000-----------------------
INT 2F U - DR DOS 6.0+ IBMBIO.COM - QUERY SIZE OF THE BDOS
  AX = 12FFh
  BX = 0000h
Return: AX = 0000h if supported
  DX = size of the BDOS in paragraphs
  Flags trashed
  ES,CL destroyed (DR PalmDOS)
Note: This API is provided by IBMBIO.COM for the initialization phase
    of drivers loaded via DEVICE=/HIDEVICE=/DEVICEHIGH= directives and
    is only available during these short time intervals.  It is called
    by DR DOS 6.0 EMM386.SYS and Novell DOS 7+ EMM386.EXE to query the
    size of the BDOS kernel (the resident code of IBMDOS.COM).
```

```
SeeAlso: AX=12FFh/BX=0001h,AX=12FFh/BX=0002h
--------O-2F12FFBX0001-----------------------
INT 2F U - DR DOS 6.0+ IBMBIO.COM - RELOCATE THE BDOS
  AX = 12FFh
  BX = 0001h
  CX = 0000h (DR PalmDOS)
  DX = segment to relocate to (FFFFh for HMA)
Return: AX = 0000h if supported
  Flags trashed
  BX,CX,DX,DI,SI,DS,ES destroyed (DR PalmDOS)
Notes:  This API is provided by IBMBIO.COM for the initialization phase
    of drivers loaded via DEVICE=/HIDEVICE=/DEVICEHIGH= directives and
    is only available during these short time intervals.  It is initiated
    by DR DOS 6.0 EMM386.SYS and Novell DOS 7+ EMM386.EXE to relocate
    the BDOS kernel (e.g. into the HMA).
  This call is also issued by DR PalmDOS IBMBIO.COM which explicitly
    clears CX.
  Under Novell DOS 7+ the actual relocation takes place at a later
    stage, but under DR PalmDOS the BDOS is relocated immediately.
SeeAlso: AX=12FFh/BX=0000h,AX=12FFh/BX=0003h
--------O-2F12FFBX0002-----------------------
INT 2F U - DR DOS 6.0+ IBMBIO.COM - QUERY SIZE OF THE BIOS
  AX = 12FFh
  BX = 0002h
Return: AX = 0000h if supported
  DX = size of the DOS BIOS in paragraphs
  CL and flags trashed
Note: This API is provided by IBMBIO.COM for the initialization phase
    of drivers loaded via DEVICE=/HIDEVICE=/DEVICEHIGH= directives and
    is only available during these short time intervals. It is called
    by DR DOS 6.0 EMM386.SYS and Novell DOS 7 EMM386.EXE to query the
    size of the DOS BIOS (the resident code of IBMBIO.COM).
SeeAlso: AX=12FFh/BX=0000h,AX=12FFh/BX=0003h
--------O-2F12FFBX0003-----------------------
INT 2F U - DR DOS 6.0+ IBMBIO.COM - RELOCATE THE BIOS
  AX = 12FFh
  BX = 0003h
  CX = 0000h (DR PalmDOS)
  DX = segment to relocate to (FFFFh for HMA)
Return: AX = 0000h if supported
  Flags trashed
```

```
Notes:  This API is provided by IBMBIO.COM for the initialization phase
    is only available during these short time intervals.  It is initiated
    by DR DOS 6.0 EMM386.SYS and Novell DOS 7 EMM386.EXE to relocate the
    resident part of the DOS BIOS.  The actual relocation takes place at
    a later stage.
  This call is also issued by DR PalmDOS IBMBIO.COM which explicitly
    clears CX.
SeeAlso: AX=12FFh/BX=0001h,AX=12FFh/BX=0002h
--------O-2F12FFBX0005-----------------------
INT 2F U - DR DOS 6.0+ - BOOT PHASE BROADCASTS FOR MEMORYMAX/RPLOADER/SECURITY
  AX = 12FFh
  BX = 0005h
  CX = 0000h
  DX = function
       0000h MemoryMAX cleanup broadcast
       0001h RPLOADER broadcast
       !!! details to follow
Note: called at three separate points inside IBMBIO.COM
--------m-2F12FFBX0006-----------------------
INT 2F U - DR DOS 6+, Novell DOS 7+ - EMM386.EXE - VIDEO MEMORY SPACE CONTROL
  AX = 12FFh
  BX = 0006h
  DX = 0000h
  CX = function
       0000h get status of video memory space (MEMMAX /V)
       0001h map memory into video memory space (MEMMAX +V)
       0002h unmap memory from video memory space (MEMMAX -V)
Return: CF clear if successful
       AX = 0000h (successful)
       BX = segment of reserved video RAM
       CX = segment of used video RAM
       DX = segment of first upper MCB
Notes:  this functionality is provided by EMM386, and partially supported by
    HIDOS.SYS
  BL specifies which program handles the call, BH is the function number
BUG:  4DOS 5.51(a) often hangs the system (reported to be reproducable), if
    MEMMAX +V is issued from the 4DOS prompt.  To avoid this, one should
    temporary load COMMAND.COM followed by MEMMAX +v , starting the
    application, MEMMAX -v and EXIT.  4DOS 5.5c does not show this
    phenomena on the same systems where 5.51a hangs.
SeeAlso: AX=D201h/BX=4849h
```

```
(Table 04105)
Values for DR DOS memory space control error code:
 00h  successful
 80h  video memory already unmapped
 81h  video memory already mapped
 82h  no video memory reserved (/VIDEO not specified)
 83h  video memory in use (graphics mode or non-standard text mode)
 84h  mapped video memory contains allocated arena(s)
 85h  hardware error
 86h  driver requires protected mode but is permanently in real mode
    ("EMM386 OFF")
--------O-2F12FFBX0007-----------------------
INT 2F U - Novell DOS 7 - SCRIPT.EXE - GET ???
  AX = 12FFh
  BX = 0007h
  CX = 0000h
Return: CF clear if installed
      AX = 0000h
      BX = ??? (4426h)
      CX = ??? (0068h)
      DX = PSP segment of resident code???
      SI = ??? (4AFAh)
      ES = resident code segment
--------O-2F12FFBX0009-----------------------
INT 2F - DR DOS 6.0+ IBMBIO.COM - REGISTER ROOT OF UPPER MEMORY LINK
  AX = 12FFh
  BX = 0009h
  DX = new value for root segment of upper memory link
Return: AX = 0000h if supported
  ES:BX modified
Notes:  the DX value is stored at offset 66h in SYSVARS (see INT 21h/AH=52h)
    and offset 18h in the Novell DOS 7 internal variable table
    (see INT 21/AX=4458h)
  This API is provided by IBMBIO.COM for the initialization phase
    of drivers loaded via DEVICE=/HIDEVICE=/DEVICEHIGH= directives and
    is only available during these short time intervals.  It is probably
    initiated by the Novell DOS 7+ EMM386.EXE.
  this function has apparently been supported since DR DOS 6.0 "Buxton"
    of 1991/03/19
  for Novell DOS 7, ES:BX points at the internal variable table DRDAT,
```

```
    but this may change in future releases
--------m-2F12FFBX0106----------------------
INT 2F U - Novell DOS 7+ - MEMORYMAX INSTALLATION CHECK
  AX = 12FFh
  BX = 0106h
Return: CF clear if successful
      AX = 0000h (successful)
      BX = EDC0h (signature "European Development Centre")
      CL = memory manager variant
    00h if HIMEM.SYS present
    01h if EMMXMA.SYS present
    02h if EMM386.EXE present (DPMI/VCPI disabled)
    03h if multitasking EMM386.EXE present (DPMI/VCPI loaded)
      CH = ??? (00h)
      DX = binary driver version, DH is major, DL is minor
      ES = segment of EMM386 device driver header (low-memory stub)
  CF set on error
      AX = 0001h
Notes:  BL specifies which program handles the call, BH is the function number
  if the word at ES:0012h is nonzero, if contains the offset within
    segment ES of the CEMM-compatible entry point (see #02617)
  if no other program has hooked INT 67, an alternate installation
    check is to test for the string
    "NOVELL 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 if it is nonzero
Index:  entry point;Novell EMM386

(Table 02617)
Call Novell EMM386.EXE entry point with:
  AH = 00h get memory manager's status???
      ???
  AH = 01h set memory manager's status???
      ???
  AH = 02h Weitek coprocessor support???
      AL = subfunction???
  more functions???
SeeAlso: #01513 at INT 21/AX=4402h/SF=02h,#03666 at INT 67/AX=FFA5h
--------m-2F12FFBX0206----------------------
INT 2F - Novell DOS 7+ - MEMORYMAX GET PAGE TABLE ENTRY
  AX = 12FFh
```

```
  BX = 0206h
  ESI = linear address
Return: CF clear if successful
      AX = 0000h (successful)
      CX = 0000h
      EDI = page table entry
  CF set on error
      AX = function status
    0000h function supported
        CX error code
           80h if linear address has no mapping
          0001h (AX > 0) function not supported
        CX undefined
Note: This function is supported by Novell DOS 7+ EMM386.EXE 3.0+ and
    possibly by HIMEM.SYS 2.3+.
SeeAlso: AX=12FFh/BX=0306h
--------m-2F12FFBX0306----------------------
INT 2F - Novell DOS 7+ - MEMORYMAX SET PAGE TABLE ENTRY
  AX = 12FFh
  BX = 0306h
  ESI = linear address
  EDI = page table entry
Return: CF clear if successful
      AX = 0000h (successful)
      CX = 0000h
  CF set on error
      AX = function status
    0000h function supported
        CX error code
           80h if linear address has no mapping
          0001h (AX > 0) function not supported
        CX undefined
Note: This function is supported by Novell DOS 7+ EMM386.EXE 3.0+ and
    possibly by HIMEM.SYS 2.3+.
SeeAlso: AX=12FFh/BX=0206h
--------m-2F12FFBX0406----------------------
INT 2F - Novell DOS 7+ - MEMORYMAX CREATE ACCESS KEY
  AX = 12FFh
  BX = 0406h
Return: CF clear if successful
      AX = 0000h (successful)
```

```
      CX = 0000h
      SI:DI = access key
  CF set on error
      AX = function status
    0000h function supported
        CX error code
           80h if access key already exists
          0001h (AX > 0) function not supported
        CX undefined
Note: This function is supported by Novell DOS 7+ EMM386.EXE 3.0+ and
    possibly by HIMEM.SYS 2.3+.
SeeAlso: AX=12FFh/BX=0506h
--------m-2F12FFBX0506----------------------
INT 2F - Novell DOS 7+ - MEMORYMAX DESTROY ACCESS KEY
  AX = 12FFh
  BX = 0506h
  SI:DI = access key
Return: CF clear if successful
      AX = 0000h (successful)
      CX = 0000h
  CF set on error
      AX = function status
    0000h function supported
        CX error code
           80h if invalid access key
           81h if no access key exists
          0001h (AX > 0) function not supported
        CX undefined
Note: This function is supported by Novell DOS 7+ EMM386.EXE 3.0+ and
    possibly by HIMEM.SYS 2.3+.
SeeAlso: AX=12FFh/BX=0406h
--------m-2F12FFBL06-------------------------
INT 2F U - Novell DOS 7 - EMM386.EXE - ???
  AX = 12FFh
  BL = 06h
  BH = function (06h-09h)
  ???
Return: CF clear if successful
  CF set on error
      AX = function status
    0000h function supported
```

```
        CX error code
          0001h (AX > 0) function not supported
        CX undefined
--------O-2F12FFBX0EDC----------------------
INT 2F U - Novell DOS 7 - EMM386.EXE - CHECK IF MULTITASKING SUPPORT LOADED???
  AX = 12FFh
  BX = 0EDCh ('EDC' = Novell European Development Center)
Return: AX = 0000h if ??? loaded
      CF clear
      BX = 0000h
Notes:  called by Novell DOS 7 TaskMgr
  if this function returns with AX=0000h, then the code necessary to
    support the API on INT 2F/AX=2780h is loaded and that API becomes
    available for use
  because the request is handled on the initial trap to the memory
    manager caused by INT instructions, this function must be invoked
    with an actual INT 2F instruction instead of some simulation such
    as a far call to the address in the interrupt vector table
SeeAlso: AX=2780h/CL=01h,AX=2780h/CL=02h,AX=2780h/CL=03h,AX=2780h/CL=04h
--------D-2F13----------------------------
INT 2F U - DOS 3.2+ - SET DISK INTERRUPT HANDLER
  AH = 13h
  DS:DX -> interrupt handler disk driver calls on read/write
  ES:BX = address to restore INT 13 to on system halt (exit from root
     shell) or warm boot (INT 19)
Return: DS:DX set by previous invocation of this function
  ES:BX set by previous invocation of this function
Notes:  IO.SYS hooks INT 13 and inserts one or more filters ahead of the
    original INT 13 handler.  The first is for disk change detection
    on floppy drives, the second is for tracking formatting calls and
    correcting DMA boundary errors, the third is for working around
    problems in a particular version of IBM's ROM BIOS
  before the first call, ES:BX points at the original BIOS INT 13; DS:DX
    also points there unless IO.SYS has installed a special filter for
    hard disk reads (on systems with model byte FCh and BIOS date
    "01/10/84" only), in which case it points at the special filter
  most DOS 3.2+ disk access is via the vector in DS:DX, although a few
    functions are still invoked via an INT 13 instruction
  during Windows 3.1 startup this function seems to be used to
    temporarily point DOS to a dummy handler in WDCTRL.386 which always
    halts the system with a fatal error message.  If DS hasn't changed
```

```
    on return from the function, Windows will display the error message
    "Invalid DOS version".
  this is a dangerous security loophole for any virus-monitoring software
    which does not trap this call ("INT13", "Nomenklatura", and many
    Bulgarian viruses are known to use it to get the original ROM entry
    point)
  the preloadable Novell DOS 7+ SECURITY.BIN driver $SECURE$ traps
    this call.
BUG:  Novell DOS 7 IBMBIO.COM before 1995-05-08 trashed AX on return from
    this function. VGACOPY by Thomas M攏kemeier's VGA Software GmbH
    crashed due to this.  Later releases of Novell DOS 7 preserved the
    contents of the AX register.
SeeAlso: INT 13/AH=01h,INT 19,INT 9D"VIRUS"
--------N-2F13-------------------------------
INT 2F U - MS-NET - ???
  AH = 13h
  ???
Return: ???
Note: supposedly used to move (or control the movement of) NCBs
--------U-2F1400-----------------------------
INT 2F C - NLSFUNC.COM - INSTALLATION CHECK
  AX = 1400h
  BX <> 0EDCh
Return: AL = status
      00h not installed, OK to install
      01h not installed, not OK to install
      FFh installed
Notes:  this function is called by the DOS v3.3+ kernel
  supported by OS/2 v1.3+ compatibility box, which always returns AL=FFh
  supported by DR DOS 5.0+ NLSFUNC v3.0+
  documented for MS-DOS 5+, but undocumented in prior versions
  DR DOS 5.0+ NLSFUNC 3.00+ returns CF set and AX=0001h, if AL was not
    00h, FEh, or FFh on entry.
SeeAlso: AX=1400h/BX=0EDCh,AX=1401h"NLSFUNC",AX=1402h"NLSFUNC"
----------2F1400BX0EDC----------------------
INT 2Fh - DR-DOS NLSFUNC 4.01+ - ENHANCED INSTALLATION CHECK
  AX = 1400h
  BX = 0EDCh
Return: AL = status
      00h not installed, OK to install
      01h not installed, not OK to install
```

```
      (for example under a multitasker)
      FFh installed
    ES:DI -> version signature ("4.01$".."4.04$" for 4.01..4.04)
  flags may be destroyed
Program: NLSFUNC 4.01+ is currently an independent project under
    development by Matthias Paul. It is not yet publically available,
    but as NLSFUNC 4.00 did, it will probably become available with
    future DR-DOS releases.
Notes:  If BX <> 0EDCh on entry, DR-DOS NLSFUNC 4.01+ performs the standard
    installation check (INT 2F/AX=1400h), and does not change ES:DI.
  DR DOS 5.0+ NLSFUNC 3.00+ returns CF set and AX=0001h, if AL was not
    00h, FEh, or FFh on entry.
  NLSFUNC 4.01+ will use the ES:DI enhancement to check the driver
    version and calculate displacements into the resident driver for
    runtime updates of internal structures like the local NLS database
    filespec, etc.
  If the returned ES points into the HMA (ES=FFFEh) care should be taken
    to actually access the HMA while checking the version signature and
    updating resident data (mutex with local A20 enable/disable).
  While previous issues of NLSFUNC installed under a multitasker,
    DR-DOS NLSFUNC 4.01+ will actually adapt to work properly in this
    environment.
SeeAlso: AX=1400h"NLSFUNC.COM",AX=14FEh,AX=14FFh,INT 21/AH=65h
--------D-2F1400-----------------------------
INT 2F - European MS-DOS 4.0 POPUP - "CheckPu" - INSTALLATION CHECK
  AX = 1400h
Return: AX = FFFFh if installed
      BX = maximum memory required to save screen and keyboard info
  CF clear if successful
  CF set on error
      AX = error code
    0002h invalid function
    0004h unknown error
Note: the POPUP interface is used by background programs (see INT 21/AH=80h)
    to communicate with the user
SeeAlso: AX=1401h"POPUP",AX=1402h"POPUP",AX=1403h"POPUP"
--------U-2F1401-----------------------------
INT 2F CU - NLSFUNC.COM - CHANGE CODE PAGE
  AX = 1401h
  DS:SI -> internal code page structure (see #02618)
  BX = new code page (see #01757 at INT 21/AX=6602h)
```

```
  DX = country code???
Return: AL = status
       00h successful
       else DOS error code
Note: this function is called by the DOS v3.3+ kernel
SeeAlso: AX=1400h"NLSFUNC",AX=1402h"NLSFUNC",INT 21/AH=66h

Format of DOS 3.30 internal code page structure:
Offset  Size  Description (Table 02618)
 00h  8 BYTEs ???
 08h 64 BYTEs name of country information file (see #02619)
 48h  WORD  system code page (see #01757 at INT 21/AX=6602h)
 4Ah  WORD  number of supported subfunctions
 4Ch  5 BYTEs data to return for INT 21/AX=6502h
 51h  5 BYTEs data to return for INT 21/AX=6504h
 56h  5 BYTEs data to return for INT 21/AX=6505h
 5Bh  5 BYTEs data to return for INT 21/AX=6506h
 60h 41 BYTEs data to return for INT 21/AX=6501h

Format of MS-DOS/PC-DOS/OS2/WinNT/PTS-DOS COUNTRY.SYS file:
Offset  Size  Description (Table 02619)
 00h  BYTE  ID tag (FFh)
 01h  7 BYTEs ASCII "COUNTRY"
 08h  8 BYTEs ??? (00h)
 10h  BYTE  ??? (01h)
 11h  BYTE  ??? (00h)
 12h  BYTE  ??? (01h)
 13h  DWORD offset of first entry in file (see #02620)
SeeAlso: #02623

Format of MS-DOS/PC-DOS/OS2/WinNT/PTS-DOS COUNTRY.SYS entry:
Offset  Size  Description (Table 02620)
 00h  WORD  number of country-codepage entries following
 02h    N Country-Codepage entries:
    Offset  Size  Description
     00h  WORD  length of entry, not counting this word (000Ch)
     02h  WORD  country ID
     04h  WORD  codepage ID
     06h  WORD  ??? (0000h)
     08h  WORD  ??? (0000h)
     0Ah  DWORD offset of country-subfunction-header in file
```

```
          (see #02621)
Notes:  multiple codepages for a country are stored consecutively
  PTS/DOS places a copyright string immediately following this structure,
    though a copyright at the end of the file is preferable
SeeAlso: #02619

Format of MS-DOS/PC-DOS/OS2/WinNT/PTS-DOS COUNTRY.SYS country-subfunc header:
Offset  Size  Description (Table 02621)
 00h  WORD  number of subfunction entries following
 02h    N subfunction entries
    Offset  Size  Description
     00h  WORD  length of subfunction entry, not counting this
          word (usually 06h)
     02h  WORD  subfunction ID
        (value passed to INT 21/AH=65h in AL)
     04h  DWORD offset within file of subfunction data entry
          (see #02622)
SeeAlso: #02620

Format of MS-DOS/PC-DOS/OS2/WinNT/PTS-DOS COUNTRY.SYS country-subfunc data::
Offset  Size  Description (Table 02622)
 00h  BYTE  ID-tag (FFh)
 01h  7 BYTEs table-type signature (blank-padded)
    "CTYINFO" general country info (subfn 01h)
    "UCASE  " uppercase table (subfn 02h)
    "LCASE  " lowercase table (subfn 03h) (DOS 6.2_)
    "FUCASE " filename uppercase table (subfn 04h)
    "FCHAR  " filename terminator table (subfn 05h)
    "COLLATE" collating sequence (subfn 06h)
    "DBCS " double-byte character table (subfn 07h)
 08h  WORD  length of following table in bytes
    (if 0000h for DBCS table, there will still be a word of 0000h)
---country info (01h)---
 0Ah  WORD  country ID (see #01400 at AH=38h)
 0Ch  WORD  code page (see #01757)
 0Eh 34 BYTEs country-dependent info (see #01399 at AH=38h)
---uppercase table (02h)---
 0Ah 128 BYTEs  uppercase equivalents (if any) of chars 80h to FFh
---lowercase table (03h)---
 0Ah 256 BYTEs  lowercase equivalents (if any) of chars 00h to FFh
---filename uppercase table (04h)---
```

```
 0Ah 128 BYTEs  uppercase equivalents (if any) of chars 80h to FFh
---filename terminator table (05h)---
 0Ah  BYTE  ??? (01h for MS-DOS 3.30-6.00)
 0Bh  BYTE  lowest permissible character value for filename
 0Ch  BYTE  highest permissible character value for filename
 0Dh  BYTE  ??? (00h for MS-DOS 3.30-6.00)
 0Eh  BYTE  first excluded character in range \ all characters in this
 0Fh  BYTE  last excluded character in range  / range are illegal
 10h  BYTE  ??? (02h for MS-DOS 3.30-6.00)
 11h  BYTE  number of illegal (terminator) characters
 12h  N BYTEs characters which terminate a filename:  ."/\[]:|<>+=;,
---collating sequence (06h)---
 0Ah 256 BYTEs  values used to sort characters 00h to FFh
---DBCS table (07h)---
 0Ah 2N BYTEs start/end for N lead byte ranges
  WORD  0000h (end of table)
SeeAlso: #02621,#01750,#01751,#01753,#01754,#01755,#01756

Format of DR DOS/Novell DOS/OpenDOS COUNTRY.SYS file:
Offset  Size  Description (Table 02623)
 00h 126 BYTEs  copyright notice (terminated with Ctrl-Z; NUL-padded)
    the copyright notice starts with the signature
    "COUNTRY.SYS Rx.xx" where "x.xx" indicates the file format
    revision, which is checked by the OS (revision is 2.00 for
    DR DOS 3.41 and 2.01 for all newer versions of DR DOS,
    Novell DOS, and OpenDOS)
 7Eh  WORD  signature of file format revision
    0EDCh = 2.00 (DR DOS 3.41)
    EDC1h = 2.01 (all newer versions)
 80h  var country pointer records (see #02624)
    (packed array of variable-size records)
SeeAlso: #02619

Format of DR DOS/Novell DOS/OpenDOS COUNTRY.SYS country pointer record::
Offset  Size  Description (Table 02624)
 00h  WORD  country code (0000h if end of array)
 02h  WORD  code page (see #01757)
 04h  WORD  ??? (0000h)
 06h  7 WORDs offsets in file for INT 21/AH=65h subfunctions 01h to 07h, or
      0000h if no table for that subfunction
 14h  var country information
```

```
Notes:  the end-of-file marker is a country pointer record filled entirely with
    zeros
  the data at which the pointers point is in the same format as the
    tables returned by INT 21/AH=65h, except that the general-info
    table for subfunction 01h does not contain the length word at the
    beginning
SeeAlso: #02623
--------D-2F1401-----------------------------
INT 2F - European MS-DOS 4.0 POPUP - "PostPu" - OPEN/CLOSE POPUP SCREEN
  AX = 1401h
  DL = function (00h open, 01h close)
  DH = wait flag
       00h block until screen opens
       01h return error if screen is not available
       02h urgent--always open screen immediately
Return: CF clear if successful
       BX = amount of memory needed to save screen and keyboard info,
    0000h if default save location can be used (only if DH was 02h)
  CF set on error
Note: the application using the screen is frozen until the popup screen is
    closed
SeeAlso: AX=1400h"POPUP",AX=1402h"POPUP",AX=1403h"POPUP"
--------U-2F1402-----------------------------
INT 2F CU - NLSFUNC.COM - GET EXTENDED COUNTRY INFO
  AX = 1402h
  BP = subfunction (same as AL for INT 21/AH=65h)
  BX = code page (see #01757 at INT 21/AX=6602h)
  DX = country code (see #01400 at INT 21/AH=38h)
  DS:SI -> internal code page structure (see #02618)
  ES:DI -> user buffer
  CX = size of user buffer
Return: AL = status
       00h successful
       else DOS error code
Notes:  this function is called by the DOS v3.3+ kernel on INT 21/AH=65h
  code page structure apparently only needed for COUNTRY.SYS pathname
SeeAlso: AX=1401h"NLSFUNC",AX=1403h"NLSFUNC",AX=1404h,INT 21/AH=65h
--------D-2F1402-----------------------------
INT 2F - European MS-DOS 4.0 POPUP - "SavePu" - SAVE POPUP SCREEN
  AX = 1402h
  ES:DI -> save buffer (0000h:0000h for default buffer in POPUP)
```

```
Return: CF clear if successful
  CF set on error
      AX = error code (see #02625)
SeeAlso: AX=1400h"POPUP",AX=1401h"POPUP",AX=1403h"POPUP"

(Table 02625)
Values for POPUP error code:
 0001h  process does not own screen
 0004h  unknown error
 0005h  invalid pointer
--------U-2F1403-----------------------------
INT 2F CU - NLSFUNC.COM - SET CODE PAGE
  AX = 1403h
  DS:SI -> internal code page structure (see #02618)
  BX = code page (see #01757 at INT 21/AX=6602h)
  DX = country code (see #01400 at INT 21/AH=38h)
Return: AL = status
       ???
Note: this function is called by the DOS v3.3+ kernel on INT 21/AH=38h
SeeAlso: AX=1402h"NLSFUNC",AX=1404h,INT 21/AH=38h"SET"
--------D-2F1403-----------------------------
INT 2F - European MS-DOS 4.0 POPUP - "RestorePu" - RESTORE SCREEN
  AX = 1403h
  ES:DI -> buffer containing saved screen
    (0000h:0000h for default buffer in POPUP)
Return: CF clear if successful
  CF set on error
      AX = error code (see #02625)
SeeAlso: AX=1400h"POPUP",AX=1401h"POPUP",AX=1402h"POPUP"
--------U-2F1404-----------------------------
INT 2F CU - NLSFUNC.COM - GET COUNTRY INFO
  AX = 1404h
  BX = code page (see #01757 at INT 21/AX=6602h)
  DX = country code (see #01400 at INT 21/AH=38h)
  DS:SI -> internal code page structure (see #02618)
  ES:DI -> user buffer
Return: AL = status
       ???
Notes:  this function is called by the DOS v3.3+ kernel on INT 21/AH=38h
  code page structure apparently only needed for COUNTRY.SYS pathname
SeeAlso: AX=1402h,AX=1403h,INT 21/AH=38h"GET"
```

```
--------U-2F14FE-----------------------------
INT 2F U - DR DOS 5.0 NLSFUNC - GET EXTENDED COUNTRY INFORMATION
  AX = 14FEh
  BX = code page (FFFFh=global code page) (see #01757 at INT 21/AX=6602h)
  DX = country ID (FFFFh=current country) (see #01400 at INT 21/AH=38h)
  ES:DI -> country information buffer
  CL = info ID
       01h get general internationalization info
       02h get pointer to uppercase table
       04h get pointer to filename uppercase table
       05h get pointer to filename terminator table
       06h get pointer to collating sequence table
       07h get pointer to Double-Byte Character Set table
  CF set (used to return error if not installed)
Return: CF clear if successful
       DS:SI -> requested information
  CF set on error
Notes:  DR DOS 5.0 NLSFUNC returns CF set and AX=0001h if AL was not 00h, FEh,
    or FFh on entry.
  the DR DOS kernel calls this function on INT 21/AX=6501h
  the value in CL is not range-checked by the DR DOS 5.0 NLSFUNC
SeeAlso: #02626,AX=14FFh,INT 21/AH=65h

Format of DR DOS COUNTRY.SYS file:
Offset  Size  Description (Table 02626)
 00h 126 BYTEs  copyright notice (terminated with Ctrl-Z, padded with NULs)
 7Eh  WORD  signature EDC1h
 80h  var country pointer records
  Offset  Size  Description
   00h  WORD  country code (0000h if end of array)
   02h  WORD  code page (see #01757 at INT 21/AX=6602h)
   04h  WORD  ??? (0000h)
   06h  7 WORDs offsets in file for data tables for subfunctions
        01h-07h
 var  var country information
--------U-2F14FF-----------------------------
INT 2F U - DR DOS 5.0+ NLSFUNC - PREPARE CODE PAGE
  AX = 14FFh
  BX = code page (see #01757 at INT 21/AX=6602h)
Return: AX = ???
  ZF set if AX=0000h
```

```
Notes:  DR DOS 5.0 NLSFUNC returns CF set and AX=0001h if AL was not 00h, FEh,
    or FFh on entry.
  passes codepage preparation request to each character device supporting
    the generic IOCTL call
BUG:  DR DOS 5.0 NLSFUNC 3.00 - Novell DOS 7 NLSFUNC 3.03, and OpenDOS 7.01 -
    DR-OpenDOS 7.02 NLSFUNC 3.02 requires DF cleared on entry, otherwise
    the system may crash.  However, since this function is called only by
    the BDOS, the problem never actually occurs.  DR-DOS 7.02/7.03
    NLSFUNC 4.00+ always clears DF by itself.
SeeAlso: AX=1400h/BX=0EDCh,AX=14FEh,INT 21/AX=440Ch,INT 21/AX=6602h
--------U-2F1500-----------------------------
INT 2F - DOS 4.00 GRAPHICS.COM - INSTALLATION CHECK
  AX = 1500h
Return: AX = FFFFh
  ES:DI -> ??? (graphics data?)
Note: this installation check conflicts with the CD-ROM Extensions
    installation check; moved to AX=AC00h in later versions
SeeAlso: AX=AC00h
--------d-2F1500BX0000-----------------------
INT 2F - CD-ROM - INSTALLATION CHECK
  AX = 1500h
  BX = 0000h
Return: BX = number of CD-ROM drive letters used
  CX = starting drive letter (0=A:)
  AX = 15FFh (Novell DOS 7 NWCDEX only!)
Notes:  this installation check DOES NOT follow the format used by other
    software
  this installation check conflicts with the DOS 4.00 GRAPHICS.COM
    installation check
BUG:  this function may return an incorrect starting drive letter when
    INTERLNK is installed
SeeAlso: AX=150Ch,AX=15FFh,INT 2F/AX=D000h"Lotus"
--------c-2F1500CH90------------------------
INT 2F U - CDBLITZ v2.11 - INSTALLATION CHECK
  AX = 1500h
  CH = 90h (function number)
  BX = 1234h (magic value for CDBLITZ)
Return: CX = 1234h if installed
      CF clear
      DX = BCD version number (DH = major, DL = minor)
Program: CDBLITZ is a CD-ROM cache by Blitz 'n' Software, Inc.
```

```
SeeAlso: AX=1500h/CH=99h
--------c-2F1500CH91-------------------------
INT 2F U - CDBLITZ v2.11 - GET STATISTICS
  AX = 1500h
  CH = 91h (function number)
  BX = 1234h (magic value for CDBLITZ)
Return: CF clear
  ES:BX -> statistics record (see #02627)
SeeAlso: AX=1500h/CH=90h,AX=1500h/CH=97h

Format of CDBLITZ statistics record:
Offset  Size  Description (Table 02627)
 00h  WORD  cache mode (see also AX=1500h/CH=94h)
    0001h 'min', 0002h 'max'
 02h  DWORD number of read calls???
 06h  DWORD total number of sectors read
 0Ah  DWORD unused??? (zero)
 0Eh  DWORD number of cache hit sectors
 12h  WORD  cache size in KB
 14h  WORD  unused??? (zero)
 16h  WORD  cache state (0000h disabled, 0001h enabled)
--------c-2F1500CH92-------------------------
INT 2F U - CDBLITZ v2.11 - ENABLE CACHE
  AX = 1500h
  CH = 92h (function number)
  BX = 1234h (magic value for CDBLITZ)
Return: CF clear
SeeAlso: AX=1500h/CH=90h,AX=1500h/CH=93h,AX=1500h/CH=94h
--------c-2F1500CH93-------------------------
INT 2F U - CDBLITZ v2.11 - DISABLE CACHE
  AX = 1500h
  CH = 93h (function number)
  BX = 1234h (magic value for CDBLITZ)
Return: CF clear
SeeAlso: AX=1500h/CH=90h,AX=1500h/CH=92h,AX=1500h/CH=95h
--------c-2F1500CH94-------------------------
INT 2F U - CDBLITZ v2.11 - SET 'MAX' MODE (CACHE BOTH DIRECTORIES AND DATA)
  AX = 1500h
  CH = 94h (function number)
  BX = 1234h (magic value for CDBLITZ)
Return: CF clear
```

```
SeeAlso: AX=1500h/CH=90h,AX=1500h/CH=92h,AX=1500h/CH=95h
--------c-2F1500CH95-------------------------
INT 2F U - CDBLITZ v2.11 - SET 'MIN' MODE (CACHE ONLY DIRECTORY ENTRIES)
  AX = 1500h
  CH = 95h (function number)
  BX = 1234h (magic value for CDBLITZ)
Return: CF clear
SeeAlso: AX=1500h/CH=90h,AX=1500h/CH=94h
--------c-2F1500CH96-------------------------
INT 2F U - CDBLITZ v2.11 - FLUSH CACHE
  AX = 1500h
  CH = 96h (function number)
  BX = 1234h (magic value for CDBLITZ)
Return: CF clear
Note: this function resets the counts for number of sectors read and number
    of cache hits, but no other values in the statistics record
    (see #02627)
SeeAlso: AX=1500h/CH=90h
--------c-2F1500CH97-------------------------
INT 2F U - CDBLITZ v2.11 - GET CACHE STATISTICS
  AX = 1500h
  CH = 97h (function number)
  BX = 1234h (magic value for CDBLITZ)
Return: CF clear
  AL = cache mode (01h 'min', 02h 'max') (see also AX=1500h/CH=94h)
  AH = cache state (00h disabled, 01h enabled)
  BX = cache size in KB
  DX:CX = total number of reads
  DI:SI = number of cache hits
SeeAlso: AX=1500h/CH=90h,AX=1500h/CH=91h
--------c-2F1500CH99-------------------------
INT 2F U - CDBLITZ v2.11 - UNINSTALL
  AX = 1500h
  CH = 99h (function number)
  BX = 1234h (magic value for CDBLITZ)
Return: CF clear
  ???
Program: CDBLITZ is a CD-ROM cache by Blitz 'n' Software, Inc.
SeeAlso: AX=1500h/CH=90h
--------d-2F1501-----------------------------
INT 2F - CD-ROM - GET DRIVE DEVICE LIST
```

```
  AX = 1501h
  ES:BX -> buffer to hold drive letter list (5 bytes per drive letter)
Return: buffer filled, for each drive letter
    BYTE  subunit number in driver
    DWORD address of device driver header (see #01646)
Note: reportedly returns AX=0000h and an invalid address under Windows95;
    other reports say it works fine
SeeAlso: AX=1510h
--------d-2F1502-----------------------------
INT 2F - CD-ROM - GET COPYRIGHT FILE NAME
  AX = 1502h
  ES:BX -> 38-byte buffer for name of copyright file
  CX = drive number (0=A:)
Return: CF set if drive is not a CD-ROM drive
      AX = 000Fh (invalid drive)
  CF clear if successful
SeeAlso: AX=1503h
--------d-2F1503-----------------------------
INT 2F - CD-ROM - GET ABSTRACT FILE NAME
  AX = 1503h
  ES:BX -> 38-byte buffer for name of abstract file
  CX = drive number (0=A:)
Return: CF set if drive is not a CD-ROM drive
      AX = 000Fh (invalid drive)
  CF clear if successful
SeeAlso: AX=1502h,AX=1504h
--------d-2F1504-----------------------------
INT 2F - CD-ROM - GET BIBLIOGRAPHIC DOC FILE NAME
  AX = 1504h
  ES:BX -> 38-byte buffer for name of bibliographic documentation file
  CX = drive number (0=A:)
Return: CF set if drive is not a CD-ROM drive
      AX = 000Fh (invalid drive)
  CF clear if successful
SeeAlso: AX=1502h,AX=1503h
--------d-2F1505-----------------------------
INT 2F - CD-ROM - READ VTOC
  AX = 1505h
  ES:BX -> 2048-byte buffer
  CX = drive number (0=A:)
  DX = sector index (0=first volume descriptor,1=second,...)
```

```
Return: CF set on error
      AX = error code (15=invalid drive,21=not ready)
  CF clear if successful
      AX = volume descriptor type (1=standard,FFh=terminator,0=other)
Note: This function was not supported by Novell DOS 7 NWCDEX prior to the
    08/16/94 update
--------d-2F1506-----------------------------
INT 2F - CD-ROM - TURN DEBUGGING ON
  AX = 1506h
  BX = debugging function to enable
Note: reserved for development
SeeAlso: AX=1507h
--------d-2F1507-----------------------------
INT 2F - CD-ROM - TURN DEBUGGING OFF
  AX = 1507h
  BX = debugging function to disable
Note: reserved for development
SeeAlso: AX=1506h
--------d-2F1508-----------------------------
INT 2F - CD-ROM - ABSOLUTE DISK READ
  AX = 1508h
  ES:BX -> buffer
  CX = drive number (0=A:)
  SI:DI = starting sector number
  DX = number of sectors to read
Return: CF set on error
      AL = error code (0Fh invalid drive,15h not ready)
  CF clear if successful
Note: returns error 15h (not ready) under Windows95 if the starting sector
    number is less than 10h
SeeAlso: AX=1509h
--------d-2F1509-----------------------------
INT 2F - CD-ROM - ABSOLUTE DISK WRITE
  AX = 1509h
  ES:BX -> buffer
  CX = drive number (0=A:)
  SI:DI = starting sector number
  DX = number of sectors to write
Note: corresponds to INT 26h and is currently reserved and nonfunctional,
    but could be implemented for CD-R and CD-RW drives
SeeAlso: AX=1508h
```

```
--------d-2F150A-----------------------------
INT 2F - CD-ROM - RESERVED
  AX = 150Ah
--------d-2F150B-----------------------------
INT 2F - CD-ROM v2.00+ - DRIVE CHECK
  AX = 150Bh
  CX = drive number (0=A:)
Return: BX = ADADh if MSCDEX.EXE installed
      AX = support status
    0000h if drive not supported
    nonzero if supported
SeeAlso: AX=150Dh
--------d-2F150CBX0000-----------------------
INT 2F - CD-ROM v2.00+ - GET MSCDEX.EXE VERSION (GET VERSION)
  AX = 150Ch
  BX = 0000h
Return: BH = major version
  BL = minor version
Notes:  MSCDEX.EXE versions prior to 2.00 leave BX unchanged, thus BX should
    be 0000h on entry
  Corel's CORELCDX.COM v1.01d returns 2.20, v1.12a returns 2.21
  Meridian Data's CDNETEX.EXE returns its own version number, e.g. 4.70
  J.M.A. Hall's CDEMU2.COM returns 2.10 (it is an MSCDEX emulator for
    networked CD-ROM drives)
  Windows95 returns v2.95
  Novell DOS 7 NWCDEX.EXE returns the same version number reported in
    its startup message
SeeAlso: AX=1500h"CD-ROM",AX=15FFh"CORELCDX"
--------d-2F150D-----------------------------
INT 2F - CD-ROM v2.00+ - GET CD-ROM DRIVE LETTERS
  AX = 150Dh
  ES:BX -> buffer for drive letter list (1 byte per drive)
Return: buffer filled with drive numbers (0=A:).  Each byte corresponds
  to the drive in the same position for function 1501h
SeeAlso: AX=150Bh
--------d-2F150E-----------------------------
INT 2F - CD-ROM v2.00+ - GET/SET VOLUME DESCRIPTOR PREFERENCE
  AX = 150Eh
  BX = subfunction
      00h get preference
    DX = 0000h
```

```
    Return: DX = preference settings
      01h set preference
    DH = volume descriptor preference
        01h = primary volume descriptor
        02h = supplementary volume descriptor
    DL = supplementary volume descriptor preference
        01h = shift-Kanji
  CX = drive number (0=A:)
Return: CF set on error
      AX = error code (15=invalid drive,1=invalid function)
  CF clear if successful
--------d-2F150F-----------------------------
INT 2F - CD-ROM v2.00+ - GET DIRECTORY ENTRY
  AX = 150Fh
  CL = drive number (0=A:)
  CH bit 0 = copy flag
    clear if direct copy
    set if copy to structure which removes ISO/High Sierra diffs
  ES:BX -> ASCIZ path name
  SI:DI -> buffer for directory entry (see #02628,#02629)
    must be 255 bytes for direct copy, 285 bytes for canonical
Return: CF set on error
      AX = error code
  CF clear if successful
      AX = disk format (0=High Sierra,1=ISO 9660)
Note: this function was not supported by Novell DOS 7 NWCDEX prior to the
    08/16/94 update

Format of CD-ROM directory entry (direct copy):
Offset  Size  Description (Table 02628)
 00h  BYTE  length of directory entry
 01h  BYTE  length of XAR in Logical Block Numbers
 02h  DWORD LBN of data, Intel (little-endian) format
 06h  DWORD LBN of data, Motorola (big-endian) format
 0Ah  DWORD length of file, Intel format
 0Eh  DWORD length of file, Motorola format
---High Sierra---
 12h  6 BYTEs date and time
 18h  BYTE  bit flags
 19h  BYTE  reserved
---ISO 9660---
```

```
 12h  7 BYTEs date and time
    (seventh byte is offset from GMT in 15-minute increments)
 19h  BYTE  bit flags
---both formats---
 1Ah  BYTE  interleave size
 1Bh  BYTE  interleave skip factor
 1Ch  WORD  volume set sequence number, Intel format
 1Eh  WORD  volume set sequence number, Motorola format
 20h  BYTE  length of file name
 21h  N BYTEs file name
  BYTE  (optional) padding if filename is odd length
      N BYTEs system data
SeeAlso: #02629,#01352

Format of CD-ROM directory entry (canonicalized):
Offset  Size  Description (Table 02629)
 00h  BYTE  length of XAR in Logical Block Numbers
 01h  DWORD Logical Block Number of file start
 05h  WORD  size of disk in logical blocks
 07h  DWORD file length in bytes
 0Bh  7 BYTEs date and time
 12h  BYTE  bit flags
 13h  BYTE  interleave size
 14h  BYTE  interleave skip factor
 15h  WORD  volume set sequence number
 17h  BYTE  length of file name
 18h 38 BYTEs ASCIZ filename
 3Eh  WORD  file version number
 40h  BYTE  number of bytes of system use data
 41h 220 BYTEs  system use data
SeeAlso: #02628
--------d-2F1510-----------------------------
INT 2F - CD-ROM v2.10+ - SEND DEVICE DRIVER REQUEST
  AX = 1510h
  CX = CD-ROM drive letter (0 = A, 1 = B, etc)
  ES:BX -> CD-ROM device driver request header (see #02597 at AX=0802h)
Return: CF clear if device driver has been called (check the request header's
        status word to determine whether an error has occurred)
      ES:BX request header updated
  CF set if device driver has not been called
      AX = error code (000Fh = invalid drive, 0001h = invalid function)
```

```
      ES:BX request header unchanged
Notes:  MSCDEX initializes the device driver request header's subunit field
    based on the drive number specified in CX
  MSCDEX v2.21 through v2.25 (at least) return error code AX=0001h if
    nested calls are attempted
BUGS: Novell DOS 7 NWCDEX prior to the 12/13/94 update did not initialize
    the subunit field
  Windows95 sets CF if CX isn't a CD-ROM drive but leaves CF unchanged
    if the drive is in fact a CD-ROM
SeeAlso: AX=0802h
--------d-2F15FFBX0000-----------------------
INT 2F - CD-ROM - CORELCDX - INSTALLATION CHECK
  AX = 15FFh
  BX = 0000h
Return: BX = ABCDh if CORELCDX loaded
Note: Corel's CORELCDX.COM is a replacement for MSCDEX.EXE; it also supports
    the standard MSCDEX installation check calls AX=1500h and AX=150Ch
SeeAlso: AX=1500h"CD-ROM",AX=150Ch
--------!---Section--------------------------
```