```
Interrupt List, part 18 of 18
Copyright (c) 1989-1999,2000 Ralf Brown
--------r-92---------------------------------
INT 92 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
--------N-92---------------------------------
INT 92 - Sangoma X.25 INTERFACE PROGRAM
  BX:DX -> control block
SeeAlso: INT 68"Sangoma"
--------e-92E1-------------------------------
INT 92 - Da Vinci eMail Dispatcher INTERFACE
  AH = E1h
  AL = function
  BX = stack count (number of words to push)
  CX:DX -> stack data (in word-reversed order ready to push)
Return: AX = status (see #03979)
Note: preserves BP, DS, SI, DI; other registers may be destroyed

(Table 03979)
Values for Da Vinci eMail function status:
 0001h  success
 FF97h  "ERS_NOT_AVAILABLE"
 FF99h  "ERS_TOO_MANY_NAMES"
 FF9Ah  "ERS_BAD_NAME_PASSWORD"
 FFE3h  "ERS_NAME_NOT_FOUND"
 FFF8h  "ERS_USE_STRING" (call NetGetError to get error string)
 FFFFh  "ERS_NO_SUCH_FILE"
--------e-92E100-----------------------------
INT 92 - Da Vinci eMail Dispatcher - "NetInitStart"
  AX = E100h
  BX = size of parameter block in words (000Ah)
  CX:DX -> parameter block (see #03980)
Return: AX = 0001h success
Desc: this function is used to initialize the dispatcher
SeeAlso: AX=E101h,AX=E103h

Format of Da Vinci eMail "NetInitStart" parameter block:
Offset  Size  Description (Table 03980)
 00h  WORD  segment of ???
 02h  WORD  offset of ???
```

```
 04h  WORD  high part of long ???
 06h  WORD  low part of long ???
 08h  WORD  high part of long ???
 0Ah  WORD  low part of long ???
 0Ch  WORD  high part of long ???
 0Eh  WORD  low part of long ???
 10h  WORD  high part of long ???
 12h  WORD  low part of long ???
--------e-92E101BX0000-----------------------
INT 92 - Da Vinci eMail Dispatcher - "NetInitCheck"
  AX = E101h
  BX = 0000h
  CX:DX ignored
Return: AX = 0001h success
SeeAlso: AX=E100h,AX=E180h
--------e-92E102BX0000-----------------------
INT 92 - Da Vinci eMail Dispatcher - "NetCheckDriver"
  AX = E102h
  BX = 0000h
  CX:DX ignored
Return: AX = 0001h success
Desc: this function is used to determine if the dispatcher is loaded
SeeAlso: AX=E10Bh,AX=E180h
--------e-92E103BX0000-----------------------
INT 92 - Da Vinci eMail Dispatcher - "NetTerminate"
  AX = E103h
  BX = 0000h
  CX:DX ignored
Return: AX = status (see #03979)
SeeAlso: AX=E100h
--------e-92E104-----------------------------
INT 92 - Da Vinci eMail Dispatcher - "NetWhereIs"
  AX = E104h
  BX = size of parameter block in words (0006h)
  CX:DX -> parameter block (see #03981)
Return: AX = status (see #03979)
Desc: this function is used to verify node address for usernames
SeeAlso: AX=E180h

Format of Da Vinci eMail "NetWhereIs" parameter block:
Offset  Size  Description (Table 03981)
```

```
 00h  WORD  segment of node address buffer
 02h  WORD  offset of node address buffer
 04h  WORD  segment of uppercase username
 06h  WORD  offset of uppercase username
 08h  WORD  segment of "DVSEMAIL"
 0Ah  WORD  offset of "DVSEMAIL"
--------e-92E105----------------------------
INT 92 - Da Vinci eMail Dispatcher - "NetOpen"
  AX = E105h
  BX = size of parameter block in words (0007h)
  CX:DX -> parameter block (see #03982)
Return: AX = 0000h Error
  AX = handle
Desc: this function is used to open a submission channel
SeeAlso: AX=E10Ah,AX=E106h,AX=E108h

Format of Da Vinci eMail "NetOpen" parameter block:
Offset  Size  Description (Table 03982)
 00h  WORD  operation (1 = read, 2 = write)
 02h  WORD  segment of uppercase To: username
 04h  WORD  offset of uppercase To: username
 06h  WORD  segment of "DVSEMAIL"
 08h  WORD  offset of "DVSEMAIL"
 0Ah  WORD  segment of node address
 0Ch  WORD  offset of node address
--------e-92E106BX0004----------------------
INT 92 - Da Vinci eMail Dispatcher - "NetRead"
  AX = E106h
  BX = 0004h
  CX:DX -> parameter block
Return: AX = 0001h
SeeAlso: AX=E108h
--------e-92E107BX0002----------------------
INT 92 - Da Vinci eMail Dispatcher - "NetGetError"
  AX = E107h
  BX = 0002h
  CX:DX -> parameter block
Return: AX = 0001h
SeeAlso: AX=E109h,AX=E180h
--------e-92E108----------------------------
INT 92 - Da Vinci eMail Dispatcher - "NetWrite"
```

```
  AX = E108h
  BX = size of parameter block in words (0004h)
  CX:DX -> parameter block (see #03983)
Return: AX = amount written
Desc: This function is used to write transactions to the dispatcher.
    The command block is written first and then another call is used
    to write the associated data.
SeeAlso: AX=E106h

Format of Da Vinci eMail "NetWrite" parameter block:
Offset  Size  Description (Table 03983)
 00h  WORD  buffer count (see #03985)
 02h  WORD  segment of command buffer (see #03984)
 04h  WORD  offset of command buffer
 06h  WORD  handle from NetOpen

Format of Da Vinci eMail command buffer:
Offset  Size  Description (Table 03984)
 00h  BYTE  command
    21h '!' Protocol commands for remote control
    41h 'A' Authorization protocol element
    42h 'B' Return(back) routing information
        Associated data is the From: username
    43h 'C' Carbon Copy list
        Associated data is a comma delimitted list of usernames
    44h 'D' Distribution list
        Associated data is a comma delimitted list of usernames
    45h 'E' Mail end marker
        No associated data
    48h 'H' Mail message header
        Associated data is a message header buffer
    4Dh 'M' Mail message
        Associated data is the body of the message
    4Fh 'O' Object
    50h 'P' Paperclip attachment
    52h 'R' Routing information
        Associated data is the To: username
    53h 'S' Subject
        Associated data is the subject of the message
    54h 'T' Trail of Reply/Forwards
 01h  BYTE  subcommand
```

```
 02h  DWORD length of associated data

Format of Da Vinci eMail message header buffer:
Offset  Size  Description (Table 03985)
 00h 30 BYTEs subject line
 1Eh 24 BYTEs To
 36h 24 BYTEs From
 4Eh  DWORD Time
    BYTE  00h
    BYTE  hour
    BYTE  minute
    BYTE  second
 52h  DWORD Date
    BYTE  00h
    BYTE  year
    BYTE  month
    BYTE  day
 56h  DWORD serial number (00000000h)
 5Ah  WORD  mail types (see #03986)
 5Ch  WORD  special types (0)

Bitfields for Da Vinci eMail mail types:
Bit(s)  Description (Table 03986)
 7  blind carbon copy
 6  carbon copy
 5  priority
 4  confidential
 3  certified
 2  bulk
 1-0  class (first, second, third, bulk)
--------e-92E109-----------------------------
INT 92 - Da Vinci eMail Dispatcher - "NetErrorFix" (UNUSED)
  AX = E109h
  BX = size of parameter block in words (0001h)
  CX:DX -> parameter block (see #03987)
Return: AX = FF97h (ERS_NOT_AVAILABLE)
SeeAlso: AX=E107h,AX=E180h

Format of Da Vinci eMail "NetErrorFix" parameter block:
Offset  Size  Description (Table 03987)
 00h  WORD  ???
```

--------e-92E10A-----------------------------
INT 92 - Da Vinci eMail Dispatcher - "NetClose"
  AX = E10Ah
  BX = size of parameter block in words (0001h)
  CX:DX -> parameter block (see #03988)
Return: AX = 0001h
Desc: this function is used to close a dispatcher handle
SeeAlso: AX=E105h

Format of Da Vinci eMail "NetClose" parameter block:
Offset Size Description (Table 03988)
 00h WORD handle from NetOpen
--------e-92E10B-----------------------------
INT 92 - Da Vinci eMail Dispatcher - "NetCheckQueue"
  AX = E10Bh
  BX = size of parameter block in words (0004h)
  CX:DX -> parameter block (see #03989)
Return: AX = 0001h
SeeAlso: AX=E102h,AX=E10Ch

Format of Da Vinci eMail "NetCheckQueue" parameter block:
Offset Size Description (Table 03989)
 00h WORD segment of 24-byte username buffer
 02h WORD offset of 24-byte username buffer
 04h WORD segment of 24-byte protocol buffer
 06h WORD offset of 24-byte protocol buffer
--------e-92E10C-----------------------------
INT 92 - Da Vinci eMail Dispatcher - "NetReadQueue"
  AX = E10Ch
  BX = size of parameter block in words (0002h)
  CX:DX -> parameter block (see #03990)
Return: AX = 0001h
SeeAlso: AX=E10Bh

Format of Da Vinci eMail "NetReadQueue" parameter block:
Offset Size Description (Table 03990)
 00h WORD Segment of 128 byte node address buffer
 02h WORD Offset of 128 byte node address buffer
--------e-92E10D-----------------------------
INT 92 - Da Vinci eMail Dispatcher - "NetSubmitName"
  AX = E10Dh

```
  BX = size of parameter block in words (0006h)
  CX:DX -> parameter block (see #03991)
Return: AX = status (see #03979)
Desc: this function is used to verify username/password
SeeAlso: AX=E10Eh

Format of Da Vinci eMail "NetSubmitName" parameter block:
Offset  Size  Description (Table 03991)
 00h  WORD  segment of uppercase password string
 02h  WORD  offset of uppercase password string
 04h  WORD  segment of uppercase username string
 06h  WORD  offset of uppercase username string
 08h  WORD  segment of "DVSEMAIL"
 0Ah  WORD  offset of "DVSEMAIL"
--------e-92E10E-----------------------------
INT 92 - Da Vinci eMail Dispatcher - "NetRemoveName"
  AX = E10Eh
  BX = size of parameter block in words (0004h)
  CX:DX -> parameter block (see #03992)
Return: AX = 0001h
Desc: this function is used to remove a username
SeeAlso: AX=E10Dh

Format of Da Vinci eMail "NetRemoveName" parameter block:
Offset  Type  Description (Table 03992)
 00h  WORD  segment of uppercase username
 02h  WORD  offset of uppercase username
 04h  WORD  segment of "DVSEMAIL"
 06h  WORD  offset of "DVSEMAIL"
--------e-92E10FBX0000-----------------------
INT 92 - Da Vinci eMail Dispatcher - IS ANYONE THERE? QUERY
  AX = E10Fh
  BX = 0000h
  CX:DX ignored
Return: AX = 0001h
SeeAlso: AX=E180h
--------e-92E110-----------------------------
INT 92 - Da Vinci eMail Dispatcher - "NetGetAltRoute"
  AX = E110h
  BX = size of parameter block in words (0006h)
  CX:DX -> parameter block (see #03993)
```

```
Return: AX = 0001h
SeeAlso: AX=E111h,AX=E113h

Format of Da Vinci eMail "NetGetAltRoute" parameter block:
Offset  Size  Description (Table 03993)
 00h  6 WORDs ???
--------e-92E111-----------------------------
INT 92 - Da Vinci eMail Dispatcher - "NetDeleteAltRoutes"
  AX = E111h
  BX = size of parameter block in words (0004h)
  CX:DX -> parameter block (see #03994)
Return: AX = 0001h
SeeAlso: AX=E110h,AX=E113h

Format of Da Vinci eMail "NetDeleteAltRoutes" parameter block:
Offset  Size  Description (Table 03994)
 00h  4 WORDs ???
--------e-92E112-----------------------------
INT 92 - Da Vinci eMail Dispatcher - "NetChangePassword"
  AX = E112h
  BX = size of parameter block in words (0008h)
  CX:DX -> parameter block (see #03995)
Return: AX = 0001h
SeeAlso: AX=E180h

Format of Da Vinci eMail "NetChangePassword" parameter block:
Offset  Size  Description (Table 03995)
 00h  8 WORDs ???
--------e-92E113-----------------------------
INT 92 - Da Vinci eMail Dispatcher - "NetSetAltRoute"
  AX = E113h
  BX = size of parameter block in words (0008h)
  CX:DX -> parameter block (see #03996)
Return: AX = 0001h
SeeAlso: AX=E110h,AX=E111h

Format of Da Vinci eMail "NetSetAltRoute" parameter block:
Offset  Size  Description (Table 03996)
 00h  8 WORDs ???
--------e-92E175-----------------------------
INT 92 - Da Vinci eMail Dispatcher - BECOME MICRO TSR
```

```
  AX = E175h
Return: AX = 0012h
  BX = PSP
SeeAlso: AX=E180h
--------e-92E180------------------------------
INT 92 - Da Vinci eMail Dispatcher - INSTALLATION CHECK
  AX = E180h
Return: AX = 0012h if installed
  ES:DX -> '$'-terminated driver information string
SeeAlso: AX=E102h,AX=E105h,AX=E10Fh,AX=E175h
--------r-93--------------------------------
INT 93 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
--------N-93--------------------------------
INT 93 - IBM TOKEN RING ADAPTER - ???
SeeAlso: INT 81"TOKEN RING",INT 91"TOKEN RING"
--------r-94--------------------------------
INT 94 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
--------s-94----SI0000----------------------
INT 94 u - PCM driver - INITIALIZE SOUND
  SI = 0000h
  ES:BX -> parameters
Return: ???
Program: PCM.COM is a sound driver for Media Vision's Pro Audio Spectrum
    sound boards
InstallCheck: test for the signature string "PCMDRIVER" immediately preceding
    the interrupt handler; the word preceding the signature gives the PCM
    driver's version
SeeAlso: SI=0001h,SI=0002h,SI=0003h,SI=0004h,SI=0005h,SI=000Ah
Index:  installation check;PCM driver|PCM.COM;installation check
Index:  PCM driver;installation check
--------s-94----SI0001----------------------
INT 94 u - PCM driver - INITIALIZE PCM
  SI = 0001h
  ES:BX -> parameters
Return: ???
SeeAlso: SI=0000h,SI=0002h,SI=0003h,SI=000Ah
--------s-94----SI0002----------------------
```

```
INT 94 u - PCM driver - INITIALIZE PCM INFO
  SI = 0002h
  ES:BX -> parameters (see #03997)
Return: ???
SeeAlso: SI=0000h,SI=0001h,SI=0003h,SI=000Ah

Format of PCM driver function 0002h parameters:
Offset  Size  Description (Table 03997)
 00h  DWORD rate
 04h  WORD  channel number
 06h  WORD  "comp"
 08h  WORD  "dsize"
--------s-94----SI0003----------------------
INT 94 u - PCM driver - INITIALIZE DMA BUFFER
  SI = 0003h
  ES:BX -> parameters (see #03998)
Return: ???
SeeAlso: SI=0000h,SI=000Ah,SI=000Bh

Format of PCM driver function 0003h parameters:
Offset  Size  Description (Table 03998)
 00h  DWORD -> DMA buffer
 04h  WORD  size of DMA buffer
 06h  WORD  number of divisions
--------s-94----SI0004----------------------
INT 94 u - PCM driver - INITIALIZE USER FUNCTION
  SI = 0004h
  ES:BX -> parameters (see #03999)
Return: ???
SeeAlso: SI=0000h,SI=0001h

Format of PCM driver function 0004h parameters:
Offset  Size  Description (Table 03999)
 00h  DWORD -> user function
--------s-94----SI0005----------------------
INT 94 u - PCM driver - BEGIN AUDIO PLAY
  SI = 0005h
Return: ???
SeeAlso: SI=0000h,SI=0006h,SI=0007h,SI=0009h
--------s-94----SI0006----------------------
INT 94 u - PCM driver - BEGIN AUDIO RECORD
```

```
  SI = 0006h
Return: ???
SeeAlso: SI=0005h,SI=0007h,SI=0009h
--------s-94----SI0007-----------------------
INT 94 u - PCM driver - PAUSE AUDIO PLAY/RECORD
  SI = 0007h
Return: ???
SeeAlso: SI=0005h,SI=0006h,SI=0008h
--------s-94----SI0008-----------------------
INT 94 u - PCM driver - RESUME AUDIO PLAY/RECORD
  SI = 0008h
Return: ???
SeeAlso: SI=0007h
--------s-94----SI0009-----------------------
INT 94 u - PCM driver - STOP AUDIO PLAY/RECORD
  SI = 0009h
Return: ???
SeeAlso: SI=0005h,SI=0006h,SI=0007h
--------s-94----SI000A-----------------------
INT 94 u - PCM driver - UNHOOK INTERRUPTS AND TURN OFF DMA
  SI = 000Ah
Return: ???
SeeAlso: SI=0000h,SI=0001h,SI=0003h
Index:  uninstall;PCM driver
--------s-94----SI000B-----------------------
INT 94 u - PCM driver - FIND VALID DMA BUFFER IN HUGE MEMORY BLOCK
  SI = 000Bh
  ES:BX -> parameters (see #04000)
Return: ???
SeeAlso: SI=0003h

Format of PCM driver functio 000Bh parameters:
Offset  Size  Description (Table 04000)
 00h  DWORD -> memory block to contain DMA buffer
 04h  WORD  desired size of DMA buffer
--------s-94----SI000D-----------------------
INT 94 u - Media Vision PCM.COM - GET STATUS
  SI = 000Dh
Return: AX = status (0000h = waiting) (see #04001)

Bitfields for PCM.COM status:
```

```
Bit(s)  Description (Table 04001)
 0  playing
 1  recording
 2  SBplaying
 3  SBrecording
 14 SBpaused
 15 paused
--------s-94----SI8000-----------------------
INT 94 u - Media Vision PCM.COM - GET INTERNAL DMA BUFFER ADDRESS
  SI = 8000h
Return: DX:AX -> DMA buffer
Program: PCM.COM is a superset of the standard PCM driver which provides
    additional functions for fine control of the driver
InstallCheck: for the Media Vision PCM.COM "shark" functions, test for the
    signature "PCM-SHARK" at offset 107h in the INT 94 handler's segment
SeeAlso: SI=8001h,SI=8004h
Index:  installation check;Media Vision PCM.COM|PCM.COM;installation check
Index:  Media Vision PCM.COM;"shark" functions
--------s-94----SI8001-----------------------
INT 94 u - Media Vision PCM.COM - GET INTERNAL DMA BUFFER SIZE AND DIVISIONS
  SI = 8001h
Return: AX = DMA buffer size
  DX = divisions
SeeAlso: SI=8000h
--------s-94----SI8002-----------------------
INT 94 u - Media Vision PCM.COM - CHECK BOARD ADDRESS
  SI = 8002h
Return: AX = status
      0000h if board not at specified I/O address
      other if board found
Note: the I/O address is specified by ORing the base I/O port shifted left
    four bits into SI before calling INT 94
SeeAlso: SI=8000h
--------s-94----SI8004-----------------------
INT 94 u - Media Vision PCM.COM - GET INTERNAL NOTE BUFFER
  SI = 8004h
Return: AX = offset of note buffer (segment = segment of internal DMA buffer)
  DX = size of buffer in note structures
SeeAlso: SI=8000h
--------s-94----SI8005-----------------------
INT 94 u - Media Vision PCM.COM - SINGLE-STEP QUEUE
```

```
  SI = 8005h
Return: ???
--------s-94----SI8011-----------------------
INT 94 u - Media Vision PCM.COM - INITIALIZE
  SI = 8011h
  ES:BX -> "iobf91" structure
Return: ???
--------s-94----SI8012-----------------------
INT 94 u - Media Vision PCM.COM - LOAD SOUND FOR LATER PLAY THROUGH KEYBOARD
  SI = 8012h
  ES:BX -> "i94f92buf" structure
Return: ???
SeeAlso: SI=8013h,SI=8014h
--------s-94----SI8013-----------------------
INT 94 u - Media Vision PCM.COM - GET INTERNAL SOUND USAGE
  SI = 8013h
Return: AX = number of sounds used
  DX = maximum handles
--------s-94----SI8014-----------------------
INT 94 u - Media Vision PCM.COM - GET DATA FOR SPECIFIED SOUND
  SI = 8014h
  ES:BX -> "i94f92buf" structure to be filled in
    sound number field set to desired sound
Return: AX = status
      0000h successful
      FFFFh sound number out of range
SeeAlso: SI=8012h,SI=8013h
--------s-94----SI8015-----------------------
INT 94 u - Media Vision PCM.COM - GET/SET INTERNAL DMA BUFFER
  SI = 8015h
  ES:BX -> DMA info structure (see #04002)
Return: ???

Format of PCM.COM DMA info structure:
Offset  Size  Description (Table 04002)
 00h  DWORD -> DMA buffer (offset FFFFh = return current buffer info)
 04h  WORD  DMA buffer size
 06h  WORD  divisions
--------s-94----SI8016-----------------------
INT 94 u - Media Vision PCM.COM - SIMULATE DOUBLE-SHIFT HOTKEY
  SI = 8016h
```

```
  AX = hotkey number (01h-08h)
Return: ???
SeeAlso: AL=02h/SI=8017h
--------s-94--01SI8017-----------------------
INT 94 u - Media Vision PCM.COM - CTRL-G INTERCEPT
  AL = 01h
  SI = 8017h
  AH = new state (00h off, 01h on)
Return: ???
SeeAlso: AL=02h/SI=8017h
--------s-94--02SI8017-----------------------
INT 94 u - Media Vision PCM.COM - DOUBLE-SHIFT-HOTKEY SOUND FEATURE
  AL = 02h
  SI = 8017h
  AH = new state (00h off, 01h on)
Return: ???
--------s-94--04SI8017-----------------------
INT 94 u - Media Vision PCM.COM - RANDOM SOUND FEATURE
  AL = 04h
  SI = 8017h
  AH = new state
       00h off
       01h on
    CX = minimum delay
    DX = maximum delay
Return: ???
--------s-94--08SI8017-----------------------
INT 94 u - Media Vision PCM.COM - NO ACTIVITY FEATURE
  AL = 08h
  SI = 8017h
  AH = new state
       00h off
       01h on
    DX:CX = delay
Return: ???
SeeAlso: AL=10h/SI=8017h
--------s-94--10SI8017-----------------------
INT 94 u - Media Vision PCM.COM - TIMER CONTROL
  AL = 10h
  SI = 8017h
  AH = timer options (see #04003)
```

```
  DX:CX = delay if AH bit 7 set (one-shot if DX bit 15 set)
Return: ???
SeeAlso: AL=08h/SI=8017h

Bitfields for PCM.COM timer options:
Bit(s)  Description (Table 04003)
 7  set timer
 6  timer active (timer turned off if clear)
 5-0  timer number
--------s-94----SI8018-----------------------
INT 94 u - Media Vision PCM.COM - GET INFO
  SI = 8018h
  AL = what to get
       00h "F92state"
       01h "F92bkgd"
       02h "I10timer"
       03h "I08state"
Return: DX:AX -> desired information
--------r-95---------------------------------
INT 95 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
--------r-95---------------------------------
INT 95 - APL*PLUS/PC - DETERMINE R= SPACE
Note: use only when the R= option is invoked on entering APL
--------r-96---------------------------------
INT 96 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
--------U-96---------------------------------
INT 96 U - KILL.COM, QKILL.COM - POP UP
Program: KILL.COM is a TSR utility that allows you to terminate programs
   by calling INT 21/AH=4Ch or reboot the computer (author unknown);
   QKILL.COM is a modification of KILL.COM by Solar Designer that
   supports QEMM's Quick Boot feature
Notes:  This interrupt is intercepted but not chained by KILL.COM; it is never
    called by KILL.COM itself. It points into the middle of KILL.COM's
    INT 09 handler and assumes specific values have been placed on the
    stack (thus it can't be called as an interrupt).
  To invoke KILL, use the following code:
    pushf
```

```
    push cs
    push offset $+0Dh
    push ax
    push es
    push 0
    pop  es
    jmp  dword ptr es:[96h*4]
--------r-97--------------------------------
INT 97 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
--------r-98--------------------------------
INT 98 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
--------r-99--------------------------------
INT 99 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
--------r-9A--------------------------------
INT 9A - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT 99,INT 9B
--------r-9B--------------------------------
INT 9B - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT 9A,INT 9C"BASIC"
--------r-9C--------------------------------
INT 9C - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT 9B,INT 9D"BASIC"
--------v-9C--------------------------------
INT 9C - VIRUS - "INT13" - ORIGINAL INT 13h VECTOR
SeeAlso: INT 8B"VIRUS",INT 9D"VIRUS",INT 9E"VIRUS",INT 9F"VIRUS"
--------r-9D--------------------------------
INT 9D - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
```

```
SeeAlso: INT 80"BASIC",INT 9C"BASIC",INT 9E"BASIC"
--------v-9D--------------------------------
INT 9D - VIRUS - "INT13" - ROM INT 13h ENTRY POINT
Note: this vector is used by the virus to store the result of a call to
    INT 2F/AH=13h
SeeAlso: INT 2F/AH=13h,INT 9C"VIRUS",INT 9E"VIRUS",INT 9F"VIRUS"
--------r-9E--------------------------------
INT 9E - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT 9D"BASIC",INT 9F"BASIC"
--------v-9E--------------------------------
INT 9E - VIRUS - "INT13" - ORIGINAL INT 21h VECTOR
SeeAlso: INT 70"VIRUS",INT 9C"VIRUS",INT 9D"VIRUS",INT E0"VIRUS"
--------r-9F--------------------------------
INT 9F - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT 9D"BASIC",INT A0"BASIC"
--------v-9F--------------------------------
INT 9F - VIRUS - "INT13" - STORAGE FOR USER INT 13h VECTOR
Note: while it is infecting a file, the INT13 virus grabs INT 13 and uses
    this interrupt to store the existing INT 13 vector for later
    restoration
SeeAlso: INT 9C"VIRUS",INT 9D"VIRUS",INT D3"VIRUS"
--------r-A0--------------------------------
INT A0 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT 9F"BASIC",INT A1"BASIC"
--------r-A0--------------------------------
INT A0 - APL*PLUS/PC - USED BY APL/GSS*CGI GRAPHICS INTERFACE
SeeAlso: INT 59"GSS"
--------r-A1--------------------------------
INT A1 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT A0"BASIC",INT A2"BASIC"
--------r-A2--------------------------------
INT A2 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
```

```
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT A1"BASIC",INT A3"BASIC"
--------r-A3---------------------------------
INT A3 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT A2"BASIC",INT A4"BASIC"
--------r-A4---------------------------------
INT A4 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT A3"BASIC",INT A5"BASIC"
--------U-A4---------------------------------
INT A4 U - Right Hand Man - API
  AH = function number (v3.3 supports functions 00h-52h)
Return: CF set on error
  CF clear if successful
Program: Right Hand Man is a TSR desk-top utility originally by Red E Products
    which has evolved into Futurus Team
Note: this interrupt is only hooked while popped up
SeeAlso: INT 2F/AX=A4E0h
--------r-A5---------------------------------
INT A5 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT A4"BASIC",INT A6"BASIC"
--------r-A6---------------------------------
INT A6 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT A5"BASIC",INT A7"BASIC"
--------r-A7---------------------------------
INT A7 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
--------r-A8---------------------------------
INT A8 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
--------r-A9---------------------------------
INT A9 - IBM ROM BASIC - used while in interpreter
```

```
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
--------r-AA---------------------------------
INT AA - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
--------r-AB---------------------------------
INT AB - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
--------r-AC---------------------------------
INT AC - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
--------r-AD---------------------------------
INT AD - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
--------r-AE---------------------------------
INT AE - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
--------r-AF---------------------------------
INT AF - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
--------r-B0---------------------------------
INT B0 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
--------r-B1---------------------------------
INT B1 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
--------r-B2---------------------------------
INT B2 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
--------r-B3---------------------------------
INT B3 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
```

```
  BASIC.COM/BASICA.COM do not restore vector on termination
--------U-B370--------------------------------
INT B3 - ZIPKEY - GET VERSION
  AH = 70h
Return: AH = major version
  AL = minor version
  CL = number of states and territories in current database
  DH = year of current database - 1900
  DL = month of current database's file date
Program: ZIPKEY is a resident ZIPCODE database by Eric Isaacson
Note: if installed, the string "ZIPKEY" is present at offset 75h in the
    interrupt handler's segment, and the byte at 7Bh contains the API
    version number (00h for v1.x, 01h for v2.0)
--------U-B371--------------------------------
INT B3 - ZIPKEY - CONVERT TWO-LETTER ABBREVIATION TO STATE CODE
  AH = 71h
  BX = abbreviation, in either case (first letter in BL)
Return: CF set on error
      AL = FFh
  CF clear if successful
      AL = ZIPKEY state code
SeeAlso: AH=72h
--------U-B372--------------------------------
INT B3 - ZIPKEY - CONVERT STATE CODE TO TWO-LETTER ABBREVIATION
  AH = 72h
  BL = ZIPKEY state code
Return: CF set on error
      AX destroyed
  CF clear if successful
      AX = abbreviation, in upper case
SeeAlso: AH=71h,AH=73h
--------U-B373--------------------------------
INT B3 - ZIPKEY - CONVERT STATE CODE TO STATE NAME
  AH = 73h
  BL = ZIPKEY state code
  ES:DI -> buffer for name
Return: CF set on error
      AX destroyed
  CF clear if successful
      ES:DI points one byte beyond end of name
SeeAlso: AH=72h
```

```
--------U-B374--------------------------------
INT B3 - ZIPKEY - CONVERT ZIPCODE TO ASCII DIGITS
  AH = 74h
  DX = zipcode region (0-999)
  CH = last two digits of zipcode (0-99)
  ES:DI -> buffer
Return: CF set on error
      AX destroyed
  CF clear if successful
      ES:DI points one byte beyond end of digit string
--------U-B375--------------------------------
INT B3 - ZIPKEY - LOOK UP STATE CODE FOR ZIPCODE
  AH = 75h
  DX = zipcode region (0-999)
  CH = last two digits of zipcode (0-99)
Return: CF set on error (zipcode not found)
      AL = suggested state code, FFh if none
  CF clear if successful
      AL = ZIPKEY state code
      BX = area code (v2.0+)
SeeAlso: AH=76h,AH=79h
--------U-B376--------------------------------
INT B3 - ZIPKEY - LOOK UP CITY AND STATE FOR ZIPCODE
  AH = 76h
  DX = zipcode region (0-999)
  CH = last two digits of zipcode (0-99)
  ES:DI -> buffer for name
Return: CF set on error
      AL = suggested state code, FFh if none
      ES:DI buffer filled with suggested city name
  CF clear if successful
      AL = ZIPKEY state code
      BX = area code (v2.0+)
      ES:DI points one byte beyond end of name
SeeAlso: AH=75h,AH=78h,AH=7Eh
--------U-B377--------------------------------
INT B3 - ZIPKEY - PLAY BACK EXIT KEY FOR ENTRY WITH GIVEN ZIPCODE
  AH = 77h
  DX = zipcode region (0-999)
  CH = last two digits of zipcode (0-99)
  BX = 16-bit BIOS keycode for a defined ZIPKEY alternate exit key
```

```
Return: CF set on error
      AX destroyed
  CF clear if successful
      zipcode specification as defined by the BX keystroke is placed in
        keyboard buffer, as if the user had popped up ZIPKEY and exited
        by pressing the key specified by BX
--------U-B378--------------------------------
INT B3 - ZIPKEY - LOOK UP ZIPCODES FOR A GIVEN STATE AND CITY
  AH = 78h
  BL = ZIPKEY state code
  DS:SI -> city name, terminated with 0Dh if complete name, 00h if prefix
Return: BH = number of matching entries (set to 51 if more than 50)
  DX = zipcode region of first match (0-999)
  CL = last two digits of first zipcode in the range (0-99)
  CH = last two digits of last zipcode in the range (0-99)
  AX destroyed
SeeAlso: AH=79h,AH=7Ah
--------U-B379--------------------------------
INT B3 - ZIPKEY - LOOK UP ZIPCODES FOR A GIVEN CITY
  AH = 79h
  BL = ZIPKEY state code of first state to search
  DS:SI -> city name, terminated with 0Dh if complete name, 00h if prefix
Return: AL = ZIPKEY state code of first matching state
  BH = number of matching entries (set to 51 if more than 50)
  DX = zipcode region of first match (0-999)
  CL = last two digits of first zipcode in first range (0-99)
  CH = last two digits of last zipcode in first range (0-99)
Note: to find all matching cities, repeat search with BL set to one more than
    the returned AL
SeeAlso: AH=78h,AH=7Ah
--------U-B37A--------------------------------
INT B3 - ZIPKEY - FETCH AN ENTRY FROM A PREVIOUS LOOKUP
  AH = 7Ah
  BL = case number (0 to one less than value returned in BH by lookup)
Return: AL = ZIPKEY state code
  DX = zipcode region (0-999)
  CL = last two digits of first zipcode in the range (0-99)
  CH = last two digits of last zipcode in the range (0-99)
SeeAlso: AH=78h,AH=79h
--------U-B37B--------------------------------
INT B3 - ZIPKEY - GET VALUES NEEDED TO SAVE ZIPKEY CONTEXT
```

```
  AH = 7Bh
Return: BL = maximum number of characters for a city name
  BH = ZIPKEY state code for last city-name search, or FFh if none
  CX:DX = internal code identifying last city search
  AX destroyed
SeeAlso: AH=7Ch
--------U-B37C-------------------------------
INT B3 - ZIPKEY - RESTORE ZIPKEY CONTEXT
  AH = 7Ch
  BL = maximum number of characters for a city name
  BH = ZIPKEY state code for last city-name search, or FFh if none
  CX:DX = internal code returned by AH=7Bh
Return: CF set on error
  CF clear if successful
  AX destroyed
SeeAlso: AH=7Bh
--------U-B37D-------------------------------
INT B3 - ZIPKEY - REQUEST POP UP
  AH = 7Dh
  BL = index number to simulate pressing a hotkey
       FFh for immediate popup with no playback on return
Return: CF set on error
       AL = error code
     FDh already busy with another request
     FEh illegal function
  CF clear if successful
       AX destroyed
       window popped up and was closed by the user
SeeAlso: AH=70h
--------U-B37E-------------------------------
INT B3 - ZIPKEY - GET NAME OF PRIMARY CITY FOR A ZIPCODE REGION
  AH = 7Eh
  DX = zipcode region (0-999)
  ES:DI -> buffer for name
Return: CF set on error
       AL = FFh region does not exist
  CF clear if successful
       AL = ZIPKEY state code
       ES:DI points one byte beyond end of name
SeeAlso: AH=76h
--------U-B37F-------------------------------
```

```
INT B3 - ZIPKEY - ENABLE/DISABLE HOTKEYS
  AH = 7Fh
  BL = function
       00h turn off hotkeys
       01h turn on hotkeys
       02h return hotkey status
       03h toggle hotkey status
Return: AL = hotkey status
       00h off
       01h on
--------U-B380--------------------------------
INT B3 - ZIPKEY v2.0+ - DETERMINE STATE FOR AREA CODE
  AH = 80h
  BX = telephone area code (decimal)
Return: CF clear if successful
       AL = ZIPKEY state code
       DX = first ZIP region for state (03E8h if Canada)
       CX = number of ZIP regions in state
  CF set on error
       AL = FFh
       DX = 03E9h
--------r-B4---------------------------------
INT B4 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
--------r-B4---------------------------------
INT B4 - StackMan - REQUEST NEW STACK
Return: SS:SP -> new stack
Program: StackMan is a freeware stack manager by Franz Veldman of ESaSS B.V.
    which functions as a replacement for the DOS STACK= command as well
    as permitting multiple TSRs to share a pool of stack space
InstallCheck: test for the string "STACKXXX" at offset 0Ah from the
    interrupt handler
SeeAlso: INT 2F/AX=C9FFh,INT B5"STACKMAN"
Index:  installation check;STACKMAN
--------r-B5---------------------------------
INT B5 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT B4"BASIC",INT B6"BASIC"
--------r-B5---------------------------------
```

```
INT B5 - StackMan - RESTORE ORIGINAL STACK
  SS:SP -> stack returned by INT B4
Return: SS:SP restored to value before INT B4
SeeAlso: INT 2F/AX=C9FFh,INT B4"StackMan"
--------m-B5--------------------------------
INT B5 U - Netroom NETSWAP4 - ???
  ???
Return: ???
SeeAlso: INT 31/AH=57h
--------r-B6--------------------------------
INT B6 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT B5"BASIC",INT B7"BASIC"
--------y-B6--------------------------------
INT B6 - (NOT A VECTOR!) - USED BY TBFENCE
Program: TBFence is a security program by ESaSS B.V. which transparently
    encrypts floppies and optionally allows only encrypted diskettes to
    be accessed
Note: the low word of this vector (0000h:02D8h) contains the segment of the
    TBFence INT 13h code, which starts with the signature word E487h;
    this forms the installation check
  the highest byte of this vector contains the start of a FAR JMP
    instruction to ???
SeeAlso: INT B7"TBFENCE"
Index:  installation check;TBFence
--------r-B7--------------------------------
INT B7 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT B6"BASIC",INT B8"BASIC"
--------y-B7--------------------------------
INT B7 - TBFENCE - ???
SeeAlso: INT B6"TBFENCE"
--------r-B8--------------------------------
INT B8 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT B7"BASIC",INT B9"BASIC"
--------r-B9--------------------------------
INT B9 - IBM ROM BASIC - used while in interpreter
```

```
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
--------r-BA--------------------------------
INT BA - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
--------r-BB--------------------------------
INT BB - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
--------r-BC--------------------------------
INT BC - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
--------r-BD--------------------------------
INT BD - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
--------r-BE--------------------------------
INT BE - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT BD"BASIC",INT BF"BASIC"
--------Q-BE--------------------------------
INT BE - DESQview/X - ???
Note: points at an IRET
SeeAlso: INT 15/AX=BFDEh/BX=0006h,INT 63"DESQview"
--------r-BF--------------------------------
INT BF - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT BE"BASIC",INT C0"BASIC"
--------r-C0--------------------------------
INT C0 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT BF"BASIC",INT C1"BASIC"
--------d-C0--------------------------------
INT C0 - AMI BIOS - DRIVE 0 DATA
Note: this vector is used by some AMI BIOSes to store the first four bytes
    of the hard disk parameter table
```

```
SeeAlso: INT 41"HARD DISK 0",INT 60"Adaptec",INT C1"AMI",INT C2"AMI"
SeeAlso: INT C3"AMI",INT C4"AMI"
--------r-C1---------------------------------
INT C1 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT C0"BASIC",INT C2"BASIC"
--------d-C1---------------------------------
INT C1 - AMI BIOS - DRIVE 0 DATA
Note: this vector is used by some AMI BIOSes to store the second four bytes
    of the hard disk parameter table
SeeAlso: INT 41"HARD DISK 0",INT 60"Adaptec",INT C0"AMI",INT C2"AMI"
SeeAlso: INT C3"AMI"
--------r-C2---------------------------------
INT C2 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT C1"BASIC",INT C3"BASIC"
--------d-C2---------------------------------
INT C2 - AMI BIOS - DRIVE 0 DATA
Note: this vector is used by some AMI BIOSes to store the third four bytes
    of the hard disk parameter table
SeeAlso: INT 41"DISK 0",INT 60"Adaptec",INT C0"AMI",INT C1"AMI",INT C3"AMI"
--------r-C3---------------------------------
INT C3 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT C2"BASIC",INT C4"BASIC"
--------d-C3---------------------------------
INT C3 - AMI BIOS - DRIVE 0 DATA
Note: this vector is used by some AMI BIOSes to store the final four bytes
    of the hard disk parameter table
SeeAlso: INT 41"DISK 0",INT 60"Adaptec",INT C0"AMI",INT C1"AMI",INT C2"AMI"
--------r-C4---------------------------------
INT C4 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT C3"BASIC",INT C5"BASIC"
--------d-C4---------------------------------
INT C4 - AMI BIOS - DRIVE 1 DATA
Note: this vector is used by some AMI BIOSes to store the first four bytes
```

```
    of the second hard disk's parameter table
SeeAlso: INT 46"HARD DISK 1",INT 64"Adaptec",INT C0"AMI",INT C5"AMI"
SeeAlso: INT C6"AMI",INT C7"AMI"
--------r-C5--------------------------------
INT C5 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT C4"BASIC",INT C6"BASIC"
--------d-C5--------------------------------
INT C5 - AMI BIOS - DRIVE 1 DATA
Note: this vector is used by some AMI BIOSes to store the second four bytes
    of the second hard disk's parameter table
SeeAlso: INT 46"HARD DISK 1",INT 64"Adaptec",INT C0"AMI",INT C4"AMI"
SeeAlso: INT C6"AMI",INT C7"AMI"
--------r-C6--------------------------------
INT C6 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT C5"BASIC",INT C7"BASIC"
--------r-C6--------------------------------
INT C6 - APL*PLUS/PC - IDENTICAL TO INT 86
Note: STSC moved its interrupts from 86h-8Ch to C6h-CCh, but did not delete
    the older interrupts
SeeAlso: INT 86"APL"
--------d-C6--------------------------------
INT C6 - AMI BIOS - DRIVE 1 DATA
Note: this vector is used by some AMI BIOSes to store the third four bytes
    of the second hard disk's parameter table
SeeAlso: INT 46"HARD DISK 1",INT 64"Adaptec",INT C0"AMI",INT C4"AMI"
SeeAlso: INT C5"AMI",INT C7"AMI"
--------r-C7--------------------------------
INT C7 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT C6"BASIC",INT C8"BASIC"
--------r-C7--------------------------------
INT C7 - APL*PLUS/PC - ???
Note: STSC moved its interrupts from 86h-8Ch to C6h-CCh, but did not delete
    the older interrupts
SeeAlso: INT 87"APL"
--------d-C7--------------------------------
```

```
INT C7 - AMI BIOS - DRIVE 1 DATA
Note: this vector is used by some AMI BIOSes to store the final four bytes
    of the second hard disk's parameter table
SeeAlso: INT 46"HARD DISK 1",INT 64"Adaptec",INT C0"AMI",INT C4"AMI"
SeeAlso: INT C5"AMI",INT C6"AMI"
--------r-C8---------------------------------
INT C8 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT C7"BASIC",INT C9"BASIC"
--------r-C8---------------------------------
INT C8 - APL*PLUS/PC - IDENTICAL TO INT 88
Note: STSC moved its interrupts from 86h-8Ch to C6h-CCh, but did not delete
    the older interrupts
SeeAlso: INT 88/AL=00h"APL",INT 88/AL=08h"APL"
--------r-C9---------------------------------
INT C9 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT C8"BASIC",INT CA"BASIC"
--------r-C9---------------------------------
INT C9 - APL*PLUS/PC - ???
Note: STSC moved its interrupts from 86h-8Ch to C6h-CCh, but did not delete
    the older interrupts
SeeAlso: INT 89"APL"
--------r-CA---------------------------------
INT CA - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT C9"BASIC",INT CB"BASIC"
--------r-CA---------------------------------
INT CA - APL*PLUS/PC - PRINT SCREEN
Note: STSC moved its interrupts from 86h-8Ch to C6h-CCh, but did not delete
    the older interrupts
SeeAlso: INT 8A"APL"
--------r-CB---------------------------------
INT CB - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT CA"BASIC",INT CC"BASIC"
--------r-CB---------------------------------
```

```
INT CB - APL*PLUS/PC - BEEP
Notes:  STSC moved its interrupts from 86h-8Ch to C6h-CCh, but did not delete
    the older interrupts
  same as printing a ^G via INT 21/AH=02h
SeeAlso: INT 8B"APL"
--------r-CC--------------------------------
INT CC - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT CB"BASIC",INT CD"BASIC"
--------r-CC--------------------------------
INT CC - APL*PLUS/PC - CLEAR SCREEN MEMORY
  AX = flag
      0000h do not save display attributes
      0001h save attributes
Note: STSC moved its interrupts from 86h-8Ch to C6h-CCh, but did not delete
    the older interrupts
SeeAlso: INT 8C"APL"
--------r-CD--------------------------------
INT CD - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT CC"BASIC",INT CE"BASIC"
--------r-CD--------------------------------
INT CD - STSC APL*PLUS/PC - MAY BE USED IN FUTURE RELEASES
--------r-CE--------------------------------
INT CE - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT CD"BASIC",INT CF"BASIC"
--------r-CE--------------------------------
INT CE - STSC APL*PLUS/PC - MAY BE USED IN FUTURE RELEASES
--------r-CF--------------------------------
INT CF - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT CE"BASIC",INT D0"BASIC"
--------r-CF--------------------------------
INT CF - APL*PLUS/PC - DEFAULT LOW-RESOLUTION TIMER FOR QUAD MF FUNCTION
SeeAlso: INT E0"APL"
--------r-D0--------------------------------
```

```
INT D0 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT CF"BASIC",INT D1"BASIC"
--------r-D0--------------------------------
INT D0 - STSC APL*PLUS/PC - MAY BE USED IN FUTURE RELEASES
--------U-D0--------------------------------
INT D0 - [not a vector!] - NJFRERAM SIGNATURE VECTOR
Program: NJFRERAM is a resident free-memory display utility by Mike "Nifty
    James" Blaszczak
Note: if NJFRERAM is installed, this vector points at the signature "NJ"
Index:  installation check;NJFRERAM
--------r-D1--------------------------------
INT D1 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT D0"BASIC",INT D2"BASIC"
--------r-D1--------------------------------
INT D1 - STSC APL*PLUS/PC - MAY BE USED IN FUTURE RELEASES
--------r-D2--------------------------------
INT D2 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT D1"BASIC",INT D3"BASIC"
--------r-D2--------------------------------
INT D2 - STSC APL*PLUS/PC - MAY BE USED IN FUTURE RELEASES
--------r-D3--------------------------------
INT D3 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT D2"BASIC",INT D4"BASIC"
--------r-D3--------------------------------
INT D3 - STSC APL*PLUS/PC - MAY BE USED IN FUTURE RELEASES
--------v-D3--------------------------------
INT D3 - VIRUS - "Antiexe" - RELOCATED INT 13
SeeAlso: INT 9F"VIRUS"
--------r-D4--------------------------------
INT D4 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT D3"BASIC",INT D5"BASIC"
```

```
--------r-D4---------------------------------
INT D4 - STSC APL*PLUS/PC - MAY BE USED IN FUTURE RELEASES
--------O-D400-------------------------------
INT D4 O - PC-MOS/386 v5.01 - OBSOLETE FUNCTIONS
  AH = 00h and 01h
Return: nothing
Desc: PC-MOS/386 v5.01 reports that these functions are no longer supported
    and enters an endless loop
Program: PC-MOS/386 is a multitasking/multiuser MS-DOS-compatible operating
    system by The Software Link, Inc.
--------O-D402-------------------------------
INT D4 - PC-MOS/386 v3.0+ - GET SYSTEM CONTROL BLOCK POINTER
  AH = 02h
Return: AX = 0000h
  ES:BX -> System Control Block in V86 mode (see #04004)
  ES:EBX -> System Control Block in native mode (see #04004)
Note: superseded by AH=26h
SeeAlso: AH=04h,AH=10h,AH=26h,AH=28h,AH=29h,AH=2Ah,INT 21/AX=3000h,INT 38

Format of PC-MOS/386 System Control Block:
Offset  Size  Description (Table 04004)
 00h  WORD  pointer to first TCB in chain
 02h 17 BYTEs reserved
 13h  WORD  pointer to current task's TCB
 15h  WORD  pointer to TCB of visible (console) task
--------O-D403-------------------------------
INT D4 - PC-MOS/386 v5.01 - GET/SET EXTENDED DIRECTORY INFORMATION
  AH = 03h
  AL = subfunction (00h get, 01h set)
  DS:(E)DX -> pathname
  ES:(E)BX -> 10-byte buffer for directory information (see #04005)
Return: CF clear if successful
      AL = permitted access level for file (00h-03h)
      ES:(E)BX -> modified buffer (AL=01h on entry)
  CF set on error
      AX = error code (see #01680 at INT 21/AH=59h/BX=0000h)
Notes:  BX/DX are used in V86 mode, EBX/EDX in native mode
  the file class cannot be changed for files because it affects the
    encryption method, but directories can have their classes changed

Format of PC-MOS/386 directory information:
```

```
Offset  Size  Description (Table 04005)
 00h  BYTE  reserved (0)
 01h  BYTE  file class ('A'-'Z' or 00h)
 02h  DWORD user ID of file creator
 06h  WORD  file creation time (see #01665 at INT 21/AX=5700h)
 08h  WORD  file creation date (see #01666 at INT 21/AX=5700h)
--------O-D404-------------------------------
INT D4 - PC-MOS/386 v3.0+ - GET TASK CONTROL BLOCK
  AH = 04h
  BX = task ID or FFFFh for calling task
Return: CF clear if successful
      ES = segment of Task Control Block (TCB) (see #04006)
  CF set on error
      AX = error code (see #01680 at INT 21/AH=59h/BX=0000h)
Note: superseded by AH=27h
SeeAlso: AH=02h,AH=27h,AH=28h,AH=29h,AH=2Ah,INT 38

Format of PC-MOS/386 Task Control Block:
Offset  Size  Description (Table 04006)
 00h  BYTE  signature byte "H" if allocated from system memory pool
 01h  BYTE  header block ID, "T" = TCB
 02h  WORD  length of block in paragraphs
 04h  WORD  segment address of next header block (0000h if last)
 06h  WORD  segment address of previous header block (0000h if first)
 08h  WORD  pointer to next TCB
 0Ah  WORD  pointer to previous TCB
 0Ch  WORD  pointer to associated TCB (if applicable)
 0Eh  WORD  reserved
---TCB---
 10h  WORD  TCB task ID
 12h  WORD  native context save area
 14h  WORD  start address of task
 16h  WORD  end address of task
 18h  BYTE  task priority
 19h  BYTE  task time slice
 1Ah  BYTE  "TCBWAIT" run status of task
 1Bh  BYTE  "TCBSTAT" what the task is waiting for
 1Ch  DWORD address of polling routine
 20h  BYTE  error code from last function call
 21h 11 BYTEs name of currently executing task
 2Ch  4 BYTEs ???
```

```
30h  BYTE  keyboard disabled if bit 1 set
31h  BYTE  current shift state and toggles
32h  2 BYTEs ???
34h  BYTE  current video mode
35h  BYTE  current video page
36h  BYTE  number of text columns per screen
37h  BYTE  number of text rows per screen
38h  WORD  length of video buffer
3Ah  WORD  video page length
3Ch  WORD  apge start address in video RAM
3Eh  4 WORDs current cursor positions for four screen pages
46h  8 BYTEs ???
4Eh  WORD  current cursor type
50h  BYTE  current palette setting
51h  BYTE  original video mode
52h  BYTE  start CRT row (00h or 01h)
53h  BYTE  video RAM in task active
54h  WORD  handle of video save area
56h  WORD  page count of video save area
58h  WORD  segment address of video save area
5Ah  WORD  poitner to first Task File Block (see #04009)
5Ch  WORD  pointer to first Current Directory Block (see #04012)
5Eh  WORD  pointer to active Current Directory Block (see #04012)
60h  BYTE  number of drives
61h  BYTE  current drive (0=A:, etc.)
62h  DWORD disk transfer address
66h  4 BYTEs ???
6Ah  BYTE  verify flag (nonzero = on)
6Bh  BYTE  break flag (nonzero = on)
6Ch  WORD  share/lock retry count
6Eh  WORD  ticks between share/lock retries
70h  BYTE  remote printer flags (see #04007)
71h  BYTE  ETX/ACK delay count
72h  WORD  spooler segment address
74h  2 BYTEs ???
76h  3 BYTEs remote printer redirection for LPT1 through LPT3 (see #04008)
79h  2 BYTEs ???
7Bh  DWORD offset of username in TCB
7Fh  BYTE  current output class
80h  7 BYTEs protection access rights, 2 bits per class (writeable!)
87h 122 BYTEs  ???
```

```
101h  BYTE  TCB sleep downcounter value
102h 20 BYTEs ???
116h  BYTE  last scan code
  ...
5D0h  DWORD far pointer to Device Driver Terminal's entry point
5D4h  WORD  offset of logical screen
5D6h  WORD  segment of logical screen
5D8h  WORD  cursor offset within page
5DAh  BYTE  screen columns
5DBh  WORD  async port number (0000h = none)
5DDh  DWORD physical baudrate
5E1h 19 BYTEs reserved for Device Driver Terminal (DDT)
  ...
7A6h  DWORD far pointer to unregister calling chain

Bitfields for PC-MOS/386 remote printer flags:
Bit(s)  Description (Table 04007)
 0  LPT1 to terminal
 1  LPT2 to terminal
 2  LPT3 to terminal
 3  escape to printer pending
 4  use XON/XOFF
 5  use ETX/ACK
 6  waiting for ACK or XON
 7  transparent printing on

(Table 04008)
Values for PC-MOS/386 remote printer redirection:
 00h  not redirected
 01h  redirected to COM1
 ...
 18h  redirected to COM24
 51h  redirected to LPT1
 52h  redirected to LPT2
 53h  redirected to LPT3

Format of PC-MOS/386 Task File Block:
Offset  Size  Description (Table 04009)
 00h  BYTE  signature byte "H" if allocated from system memory pool
 01h  BYTE  header block ID, "F" = task file block
 02h  WORD  length of block in paragraphs
```

```
 04h  WORD  segment address of next header block (0000h if last)
 06h  WORD  segment address of previous header block (0000h if first)
 08h  WORD  pointer to next TCB
 0Ah  WORD  pointer to previous TCB
 0Ch  WORD  pointer to associated TCB (if applicable)
 0Eh  WORD  reserved
---TFB---
 10h  WORD  segment address of next TFB
 12h  WORD  segment address of previous TFB
 14h  WORD  segment address of TFB's Global File Block (see #04011)
 16h  WORD  segment address of owner's PSP
 18h  WORD  file handle
 1Ah  3 BYTEs ???
 1Dh  DWORD file position
 21h  4 BYTEs ???
 25h  BYTE  IOCTL flags (see #04010)
 26h  2 BYTEs ???

Bitfields for PC-MOS/386 IOCTL flags:
Bit(s)  Description (Table 04010)
 0  stdin
 1  stdout
 2  null device
 3  clock device
 4  reserved
 5  ASCII mode instead of binary
 6  EOF encountered on input
 7  device rather than file

Format of PC-MOS/386 Global File Block:
Offset  Size  Description (Table 04011)
 00h  BYTE  signature byte "H" if allocated from system memory pool
 01h  BYTE  header block ID, "G" = global file block
 02h  WORD  length of block in paragraphs
 04h  WORD  segment address of next header block (0000h if last)
 06h  WORD  segment address of previous header block (0000h if first)
 08h  WORD  pointer to next TCB
 0Ah  WORD  pointer to previous TCB
 0Ch  WORD  pointer to associated TCB (if applicable)
 0Eh  WORD  reserved
---GFB---
```

```
 10h 10 BYTEs ???
 1Ah  WORD  file attribute
 1Ch  BYTE  ???
 1Dh  DWORD address of device driver
 21h  WORD  first cluster
 23h  WORD  time of last modification
 25h  WORD  date of last modification
 27h  DWORD size of file in bytes
 2Bh 11 BYTEs ???
 36h 11 BYTEs device name or FCB-format filename
 41h  WORD  segment address of TFB list
 43h  WORD  segment address of first RLB (see #04014) (0000h = none)
 45h  BYTE  flag: nonzero if GFB refers to character device
 46h  WORD  address of Block Device Block (see #04013)
 48h  WORD  sector of file's directory entry (see #01352)
 4Ah  WORD  high word of file's directory entry
 4Ch  WORD  ofsset of directory entry within sector

Format of PC-MOS/386 Current Directory Block:
Offset  Size  Description (Table 04012)
 00h  BYTE  signature byte "H" if allocated from system memory pool
 01h  BYTE  header block ID, "C" = current directory block
 02h  WORD  length of block in paragraphs
 04h  WORD  segment address of next header block (0000h if last)
 06h  WORD  segment address of previous header block (0000h if first)
 08h  WORD  pointer to next TCB
 0Ah  WORD  pointer to previous TCB
 0Ch  WORD  pointer to associated TCB (if applicable)
 0Eh  WORD  reserved
---CDB---
 10h  BYTE  drive number
 11h  BYTE  ???
 12h 64 BYTEs directory name
 52h  WORD  first directory cluster (0000h = root)

Format of PC-MOS/386 Block Device Block:
Offset  Size  Description (Table 04013)
 00h  BYTE  signature byte "H" if allocated from system memory pool
 01h  BYTE  header block ID, "B" = block device block
 02h  WORD  length of block in paragraphs
 04h  WORD  segment address of next header block (0000h if last)
```

```
 06h  WORD  segment address of previous header block (0000h if first)
 08h  WORD  pointer to next TCB
 0Ah  WORD  pointer to previous TCB
 0Ch  WORD  pointer to associated TCB (if applicable)
 0Eh  WORD  reserved
---BDB---
 10h  BYTE  logical drive
 11h  BYTE  unit passed to driver
 12h  WORD  sector size
 14h  BYTE  cluster mask
 15h  BYTE  cluster shift count
 16h  WORD  starting sector of first FAT
 18h  BYTE  number of FATs
 19h  WORD  number of root directories
 1Bh  WORD  sector number of first data sector (cluster 0002h)
 1Dh  WORD  number of clusters + 1 (number of highest data cluster)
 1Fh  BYTE  number of sectors in FAT
 20h  WORD  beginning root directory sector number
 22h  DWORD device driver address
 26h  BYTE  media descriptor byte
 27h  5 BYTEs ???
 2Ch  BYTE  flag: volume > 32MB
 2Dh  BYTE  ???
 2Eh  BYTE  number of sectors per cluster
 2Fh  WORD  number of clusters on device
 31h  WORD  number of free clusters (FFFFh = unknown)
 33h  WORD  root directory cluster number
 35h  WORD  pointer to alias/subst string
 37h  WORD  TCB segment address of owner (0000h = none)

Format of PC-MOS/386 Record Lock Block:
Offset  Size  Description (Table 04014)
 00h  BYTE  signature byte "H" if allocated from system memory pool
 01h  BYTE  header block ID, "R" = record lock block
 02h  WORD  length of block in paragraphs
 04h  WORD  segment address of next header block (0000h if last)
 06h  WORD  segment address of previous header block (0000h if first)
 08h  WORD  pointer to next TCB
 0Ah  WORD  pointer to previous TCB
 0Ch  WORD  pointer to associated TCB (if applicable)
 0Eh  WORD  reserved
```

```
---RLB---
 10h  WORD  segment address of owner's PSP
 12h  WORD  segment address of Global File Block (see #04011)
 14h  WORD  segment address of owner's Task File Block (see #04009)
 16h  DWORD file offset of locked region start
 1Ah  DWORD length of locked region
 1Eh  WORD  owner's handle for file
--------O-D407-------------------------------
INT D4 - PC-MOS/386 v3.0+ - WAIT FOR EVENT
  AH = 07h
  AL = events to monitor (see #04015)
  BX = number of timer ticks until timeout if AL bit 1 set
  CX = bitmap of IRQs to monitor if AL bit 2 set
    (bit 0 = IRQ0 .. bit 15 = IRQ15)
  DX = port to monitor if AL bit 3 set
Return: CF clear if successful
      AL = type of event which woke up task (see #04015)
      CX = IRQ (if any) which awakened task
      DX = port (if any) which awakened task
  CF set on error
      AX = error code (see #01680 at INT 21/AH=59h/BX=0000h)
Note: a device driver may make this call with AL=00h, which indicates that
    the driver is responsible for setting and clearing the TCBWAIT field
    in the TCB.  To put task to sleep, set TCBWAIT bits 2-0 to 001; to
    reawaken it, set bit 1 (leaving other bits unchanged)
SeeAlso: AH=04h,INT 16/AH=00h,INT 38

Bitfields for PC-MOS/386 events to monitor:
Bit(s)  Description (Table 04015)
 0  keystroke
 1  timeout
 2  IRQ
 3  port access
 7  return status of user poll routine (other ignored if set)
--------O-D410-------------------------------
INT D4 - PC-MOS/386 v3.0+ - ENTER/LEAVE NATIVE 386 EXECUTION MODE
  AH = 10h
  AL = direction (00h return to V86 mode, 01h enter native mode)
  CX = length in bytes of Native Context Area ( >=1024 )
  DX = segment of Native Context Area
Return: CF clear if successful
```

```
      running in desired mode at instruction following INT D4 call
      all segment registers converted to appropriate selectors/segments
  CF set on error
      AX = error code (see #01680 at INT 21/AH=59h/BX=0000h)
Note: MS-DOS calls are available in protected mode
SeeAlso: AH=11h,AH=12h,AH=13h,INT 2F/AX=1687h,INT 67/AX=DE0Ch,INT 38
--------O-D411-------------------------------
INT D4 - PC-MOS/386 v3.0+ - ALLOCATE NATIVE MODE MEMORY BLOCK
  AH = 11h
  EBX = block length in bytes
Return: CF clear if successful
      EBX = number of bytes actually allocated
      ES = selector for allocated block
  CF set on error
      AX = error code (see #01680 at INT 21/AH=59h/BX=0000h)
Program: PC-MOS/386 is a multitasking/multiuser MS-DOS-compatible operating
    system by The Software Link, Inc.
Note: the memory must be released before the program terminates
SeeAlso: AH=10h,AH=12h,INT 38
--------O-D412-------------------------------
INT D4 - PC-MOS/386 v3.0+ - FREE NATIVE MODE MEMORY BLOCK
  AH = 12h
  ES = selector for block to free
Return: CF clear if successful
  CF set on error
      AX = error code (see #01680 at INT 21/AH=59h/BX=0000h)
SeeAlso: AH=10h,AH=11h,AH=13h,INT 38
--------O-D413-------------------------------
INT D4 - PC-MOS/386 v5.01 - GET ALIAS FOR SELECTOR (NATIVE MODE ONLY)
  AH = 13h
  AL = type of alias selector (00h data, 01h stack, 02h code)
  BX = selector
Return: CF clear if successful
      AX = new selector or 0000h if BX selector not found
  CF set on error
      AX = error code (see #01680 at INT 21/AH=59h/BX=0000h)
SeeAlso: AH=11h,AH=12h
--------O-D416-------------------------------
INT D4 - PC-MOS/386 v5.01 - SET/CLEAR IRQ RESERVATION
  AH = 16h
  AL = function (00h clear, 01h set reservation)
```

```
  CX = IRQ number
Return: AX = status
      (0000h successful, 0001h currently reserved by another task)
SeeAlso: AH=07h,INT 14/AH=11h"PC-MOS"
--------O-D419------------------------------
INT D4 - PC-MOS/386 v5.01 - GET TASK ID
  AH = 19h
Return: BX = caller's task ID
Program: PC-MOS/386 is a multitasking/multiuser MS-DOS-compatible operating
    system by The Software Link, Inc.
SeeAlso: AH=1Dh,AH=1Eh
--------O-D41A------------------------------
INT D4 - PC-MOS/386 v5.01 - GET/SET TASK PRIORITY
  AH = 1Ah
  AL = subfunction (00h read, 01h set, 02h get and set)
  BX = task ID (FFFFh for current task)
  CL = new priority value
Return: CF clear if successful
      CL = current priority value
  CF set on error
      AX = error code (see #01680 at INT 21/AH=59h/BX=0000h)
SeeAlso: AH=1Bh,AH=1Ch
--------O-D41B------------------------------
INT D4 - PC-MOS/386 v5.01 - GET/SET TIME SLICE
  AH = 1Bh
  AL = subfunction (00h read, 01h set, 02h get and set)
  BX = task ID (FFFFh for current task) (see AH=19h)
  CL = new time slice value
Return: CF clear if successful
      CL = current time slice value
  CF set on error
      AX = error code (see #01680 at INT 21/AH=59h/BX=0000h)
SeeAlso: AH=1Ah,AH=1Ch
--------O-D41C------------------------------
INT D4 - PC-MOS/386 v5.01 - GET/SET KEYBOARD MODE
  AH = 1Ch
  AL = subfunction (00h enable, 01h disable, 02h get mode)
  BX = task ID (FFFFh for current task)
Return: CF clear if successful
      CL = current keyboard state
  CF set on error
```

```
      AX = error code (see #01680 at INT 21/AH=59h/BX=0000h)
Program: PC-MOS/386 is a multitasking/multiuser MS-DOS-compatible operating
    system by The Software Link, Inc.
SeeAlso: AH=1Ah,AH=1Bh
--------O-D41D---------------------------------
INT D4 - PC-MOS/386 v5.01 - GET CURRENT PROGRAM NAME
  AH = 1Dh
  BX = task ID (FFFFh for current task) (see AH=19h)
  ES:DI -> buffer for program name (see #04016)
Return: CF clear if successful
      ES:DI buffer filled
  CF set on error
      AX = error code (see #01680 at INT 21/AH=59h/BX=0000h)
SeeAlso: AH=19h,AH=1Eh

Format of PC-MOS/386 program name buffer:
Offset  Size  Description (Table 04016)
 00h   8 BYTEs filename
 08h   3 BYTEs extension
--------O-D41E---------------------------------
INT D4 - PC-MOS/386 v5.01 - GET CURRENT USERNAME AND SECURITY CLASS
  AH = 1Eh
  BX = task ID (FFFFh for current task)
  ES:DI -> 4-byte buffer for username
Return: CF clear if successful
      CL = security class
    20h (' ') none
    41h-5Ah ('A'-'Z') security level
      ES:DI buffer filled
  CF set on error
      AX = error code (see #01680 at INT 21/AH=59h/BX=0000h)
SeeAlso: AH=19h,AH=1Dh
--------O-D41F---------------------------------
INT D4 - PC-MOS/386 v5.01 - GET TASK PARTITION INFORMATION
  AH = 1Fh
  BX = task ID (FFFFh for current task) (see AH=19h)
Return: CF clear if successful
      CX = start segment of task
      DX = ending segment of task
  CF set on error
      AX = error code (see #01680 at INT 21/AH=59h/BX=0000h)
```

```
Program: PC-MOS/386 is a multitasking/multiuser MS-DOS-compatible operating
    system by The Software Link, Inc.
SeeAlso: AH=2Dh
--------O-D420---------------------------------
INT D4 - PC-MOS/386 v5.01 - GET PORT AND BAUDRATE INFORMATION
  AH = 20h
  BX = task ID (FFFFh for current task) (see AH=19h)
Return: CF clear if successful
      CX = port number (0000h if none)
      DI:SI = baudrate (if CX nonzero)
  CF set on error
      AX = error code (see #01680 at INT 21/AH=59h/BX=0000h)
SeeAlso: INT 14/AH=0Ch"FOSSIL"
--------O-D421---------------------------------
INT D4 - PC-MOS/386 v5.01 - REMOVE A TASK
  AH = 21h
  BX = task ID (FFFFh for current task) (see AH=19h)
Return: CF clear if successful
      AX = ASCII percentage of System Memory Pool used
    (AH = tens digit, AL = ones digit)
      DS,SI destroyed
  CF set on error
      AX = error code (see #01680 at INT 21/AH=59h/BX=0000h)
Program: PC-MOS/386 is a multitasking/multiuser MS-DOS-compatible operating
    system by The Software Link, Inc.
SeeAlso: AH=22h
--------O-D422---------------------------------
INT D4 - PC-MOS/386 v5.01 - ADD A TASK TO THE SYSTEM
  AH = 22h
  DS:SI -> addtask data structure (see #04018)
Return: CF clear if successful
      ES = segment address of the new task's TCB data structure
  CF set on error
      AX = error code (see #04017)
SeeAlso: AH=21h

(Table 04017)
Values for PC-MOS/386 error code:
 08h  insufficient memory
 0Bh  invalid addtask structure format
 12h  insufficient available space in system memory pool
```

```
 1Fh  general failure
 55h  already allocated
 57h  if task already in use or invalid parameter

Format of PC-MOS/386 addtask data structure:
Offset  Size  Description (Table 04018)
 00h  WORD  task size in KB (min 16KB)
 02h  WORD  task ID (0000h for automatic selection)
 04h  BYTE  task class (' ' or 'A'-'Z')
 05h  DWORD -> ASCIZ name of task startup batchfile
 09h  DWORD -> task's terminal driver (0000000h = background task)
 0Dh  WORD  task port
 0Fh  DWORD task baud rate
 13h  DWORD (ret) total extended memory
 17h  DWORD (ret) number of 4K extended memory pages allocated
 1Bh  WORD  (ret) paragraphs of system memory pool allocated
 1Dh  WORD  (ret) system memory pool size in paragraphs
 1Fh  WORD  (ret) ASCII task percentage of system memory pool
 21h  3 BYTEs reserved
--------O-D423--------------------------------
INT D4 - PC-MOS/386 v5.01 - CHANGE TERMINAL DRIVER
  AH = 23h
  BX = task ID (FFFFh for current task)
  DS:SI -> entry point of the new Device Driver Terminal
Return: CF clear if successful
  CF set on error
      AX = error code (see #01680 at INT 21/AH=59h/BX=0000h)
--------O-D424--------------------------------
INT D4 U - PC-MOS/386 v5.01 - GET OPERATING SYSTEM SERIAL NUMBER
  AH = 24h
Return: DS:DX -> '$'-terminated string containing the serial number
--------O-D425--------------------------------
INT D4 - PC-MOS/386 v5.01 - IDENTIFY LOAD ADDRESS OF DEVICE DRIVER LOCATION
  AH = 25h
  DX = driver's CS value
Return: AX = segment address of driver in system memory pool
      (0000h if the driver is not within the system memory pool)
Program: PC-MOS/386 is a multitasking/multiuser MS-DOS-compatible operating
    system by The Software Link, Inc.
--------O-D426--------------------------------
INT D4 - PC-MOS/386 v5.01 - GET SYSTEM CONTROL BLOCK SEGMENT/SELECTOR
```

```
  AH = 26h
Return: DX = segment/selector of the System Control Block (see #04004)
Note: this function supersedes AH=02h
SeeAlso: AH=02h,AH=27h,AH=28h,AH=29h,AH=2Ah
--------O-D427-------------------------------
INT D4 - PC-MOS/386 v5.01 - GET TASK CONTROL BLOCK SEGMENT/SELECTOR
  AH = 27h
  BX = task ID (FFFFh if current task) (see AH=19h)
Return: CF clear if successful
      DX = segment/selector for the Task Control Block (see #04006)
  CF set on error
      AX = error code (see #01680 at INT 21/AH=59h/BX=0000h)
Note: this function supersedes AH=04h
SeeAlso: AH=26h,AH=28h,AH=29h,AH=2Ah
--------O-D428-------------------------------
INT D4 - PC-MOS/386 v5.01 - GET CONTROL BLOCK DATA FROM SCB OR TCB
  AH = 28h
  BX = offset into control block at which to start reading
  CX = number of bytes to read
  DX = segment/selector of control block obtained via AH=26h or AH=27h
  ES:DI -> buffer for data
Return: CF clear if successful
  CF set on error
      AX = error code (see also #01680 at INT 21/AH=59h/BX=0000h)
    05h access denied due to an invalid segment/selector
SeeAlso: AH=26h,AH=27h,AH=29h,AH=2Ah
--------O-D429-------------------------------
INT D4 - PC-MOS/386 v5.01 - WRITE CONTROL BLOCK DATA INTO SCB OR TCB
  AH = 29h
  BX = offset into control block at which to start writing
  CX = number of bytes to write
  DX = segment/selector of control block obtained via AH=26h or AH=27h
  DS:SI -> buffer containing data to be written
Return: CF clear if successful
  CF set on errro
      AX = error code (see also #01680 at INT 21/AH=59h/BX=0000h)
    05h access denied due to an invalid segment/selector
Note:  this function performs no bounds checking
Program: PC-MOS/386 is a multitasking/multiuser MS-DOS-compatible operating
    system by The Software Link, Inc.
SeeAlso: AH=26h,AH=27h,AH=28h,AH=2Ah
```

```
--------O-D42A--------------------------------
INT D4 - PC-MOS/386 v5.01 - SWAP CONTROL BLOCK DATA OF SCB OR TCB
  AH = 2Ah
  BX = offset into control block at which to start swap
  CX = number of bytes to swap
  DX = segment/selector of control block obtained via AH=26h or AH=27h
  DS:SI -> buffer containing new data and to receive current data
Return: CF clear if successful
  CF set on error
      AX = error code (see also #01680 at INT 21/AH=59h/BX=0000h)
    05h access denied due to an invalid segment/selector
Note: the interrupts are disabled during the swap to prevent corruption
SeeAlso: AH=26h,AH=27h,AH=28h,AH=29h
--------O-D42C--------------------------------
INT D4 - PC-MOS/386 v5.01 - GET/SET SPOOLER PARAMETERS
  AH = 2Ch
  AL = function
      00h set spooler timeout
    CX = timout value in seconds
      01h get spooler timeout
    Return: CX = current timeout in seconds
      02h get spooler parameters
    Return: CH = priority (00h-09h)
      CL = disposition (d, h, i, n, s)
      SI = class (a - z)
      03h set spooler parameters
    CH = priority (00h-09h)
    CL = disposition (d, h, i, n, s)
    SI = class (a - z)
    BX = task ID (FFFFh for current task)
    DX = LPT number
Return: CF clear if successful
  CF set on error
      AX = error code (see #01680 at INT 21/AH=59h/BX=0000h)
SeeAlso: AH=00h,AH=02h,AH=03h
--------O-D42D--------------------------------
INT D4 - PC-MOS/386 v5.01 - GET MAXIMUM TASK SIZE
  AH = 2Dh
Return: DX = maximum task size in paragraphs
  BX = start address of task space
Program: PC-MOS/386 is a multitasking/multiuser MS-DOS-compatible operating
```

```
    system by The Software Link, Inc.
SeeAlso: AH=1Fh,AH=22h
--------r-D5--------------------------------
INT D5 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT D4"BASIC",INT D6"BASIC"
--------r-D5--------------------------------
INT D5 - STSC APL*PLUS/PC - MAY BE USED IN FUTURE RELEASES
--------r-D6--------------------------------
INT D6 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT D5"BASIC",INT D7"BASIC"
--------r-D6--------------------------------
INT D6 - STSC APL*PLUS/PC - MAY BE USED IN FUTURE RELEASES
--------r-D7--------------------------------
INT D7 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT D6"BASIC",INT D8"BASIC"
--------r-D7--------------------------------
INT D7 - STSC APL*PLUS/PC - MAY BE USED IN FUTURE RELEASES
--------r-D8--------------------------------
INT D8 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT D7"BASIC",INT D9"BASIC"
--------r-D8--------------------------------
INT D8 - STSC APL*PLUS/PC - MAY BE USED IN FUTURE RELEASES
--------H-D8--------------------------------
INT D8 - Screen Thief v1.00 - RELOCATED IRQ0
Range:  INT 78h to INT E0h, selected by commandline switch
Note: Screen Thief relocates IRQs 0 through 7 to INT D8 to INT DF by default,
    but may be directed via a commandline switch to use any range
    starting at a multiple of 8 between 78h and E0h
SeeAlso: INT 08"IRQ0",INT 2D/AL=10h"Screen Thief",INT 50"DESQview"
SeeAlso: INT D9"Screen Thief"
--------r-D9--------------------------------
INT D9 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
```

```
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT D8"BASIC",INT DA"BASIC"
--------r-D9--------------------------------
INT D9 - STSC APL*PLUS/PC - MAY BE USED IN FUTURE RELEASES
--------H-D9--------------------------------
INT D9 - Screen Thief v1.00 - RELOCATED IRQ1
Range:  INT 79h to INT E1h, selected by commandline switch
Note: (see INT D8"Screen Thief")
SeeAlso: INT 09"IRQ1",INT D8"Screen Thief",INT DA"Screen Thief"
--------r-DA--------------------------------
INT DA - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT D9"BASIC",INT DB"BASIC"
--------r-DA--------------------------------
INT DA - STSC APL*PLUS/PC - MAY BE USED IN FUTURE RELEASES
--------H-DA--------------------------------
INT DA - Screen Thief v1.00 - RELOCATED IRQ2
Range:  INT 7Ah to INT E2h, selected by commandline switch
Note: (see INT D8"Screen Thief")
SeeAlso: INT 0A"IRQ2",INT D9"Screen Thief",INT DB"Screen Thief"
--------r-DB--------------------------------
INT DB - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT DA"BASIC",INT DC"BASIC"
--------r-DB--------------------------------
INT DB - STSC APL*PLUS/PC - MAY BE USED IN FUTURE RELEASES
--------H-DB--------------------------------
INT DB - Screen Thief v1.00 - RELOCATED IRQ3
Range:  INT 7Bh to INT E3h, selected by commandline switch
Note: (see INT D8"Screen Thief")
SeeAlso: INT 0B"IRQ3",INT DA"Screen Thief",INT DC"Screen Thief"
--------u-DC--------------------------------
INT DC - PC/370 v4.1- - API
SeeAlso: INT 60"PC/370"
--------r-DC--------------------------------
INT DC - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT DB"BASIC",INT DD"BASIC"
```

```
--------r-DC--------------------------------
INT DC - STSC APL*PLUS/PC - MAY BE USED IN FUTURE RELEASES
--------H-DC--------------------------------
INT DC - Screen Thief v1.00 - RELOCATED IRQ4
Range:  INT 7Ch to INT E4h, selected by commandline switch
Note: (see INT D8"Screen Thief")
SeeAlso: INT 0C"IRQ4",INT DB"Screen Thief",INT DD"Screen Thief"
--------r-DD--------------------------------
INT DD - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT DC"BASIC",INT DE"BASIC"
--------r-DD--------------------------------
INT DD - STSC APL*PLUS/PC v9.0 - PLACE KEYSTROKE EVENTS IN INPUT BUFFER
  BX = where to place keystrokes
       FFFFh insert before current buffer contents
       0000h replace current contents
       0001h insert after current contents
  CX = number of keystroke events to insert
  ES:SI -> data to be placed into buffer (list of WORD key codes)
       4000h + N = normal ASCII keystroke N (N = 00h to FFh)
       4100h + N = extended ASCII keystroke N (N = 03h to 84h)
Return: nothing
SeeAlso: INT 16/AH=05h
--------H-DD--------------------------------
INT DD - Screen Thief v1.00 - RELOCATED IRQ5
Range:  INT 7Dh to INT E5h, selected by commandline switch
Note: (see INT D8"Screen Thief")
SeeAlso: INT 0D"IRQ5",INT DC"Screen Thief",INT DE"Screen Thief"
--------r-DE--------------------------------
INT DE - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT DD"BASIC",INT DF"BASIC"
--------r-DE--------------------------------
INT DE - APL*PLUS/PC - ???
Note: appears to be the same as INT 16
--------H-DE--------------------------------
INT DE - Screen Thief v1.00 - RELOCATED IRQ6
Range:  INT 7Eh to INT E6h, selected by commandline switch
Note: (see INT D8"Screen Thief")
```

```
SeeAlso: INT 0E"IRQ6",INT DD"Screen Thief",INT DF"Screen Thief"
--------b-DF--------------------------------
INT DF - Victor 9000/Sirius 1 - SuperBIOS
SeeAlso: INT 21/AH=EAh"NetWare"
--------r-DF--------------------------------
INT DF - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT DE"BASIC",INT E0"BASIC"
--------r-DF--------------------------------
INT DF - APL*PLUS/PC - SAME AS INT 10
SeeAlso: INT 10/AH=00h,INT 10/AH=0Eh
--------H-DF--------------------------------
INT DF - Screen Thief v1.00 - RELOCATED IRQ7
Range:  INT 7Fh to INT E7h, selected by commandline switch
Note: (see INT D8"Screen Thief")
SeeAlso: INT 0F"IRQ7",INT DE"Screen Thief"
--------r-E0--------------------------------
INT E0 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT 80"BASIC",INT DF"BASIC",INT E1"BASIC"
--------r-E0--------------------------------
INT E0 - APL*PLUS/PC - RESTIME HIGH-RESOLUTION TIMER FOR QUAD MF FUNCTION
SeeAlso: INT CF"APL"
--------v-E0--------------------------------
INT E0 - VIRUS - "Micro-128" - ???
Note: Micro-128 also overwrites the upper half of the interrupt table
SeeAlso: INT 9E"VIRUS",INT F1"VIRUS"
----------E0--------------------------------
INT E0 - DeskMate (Tandy) - DESK EXECUTIVE API
  AX = function code (numerous)
  parameters passed in BX, DX, ES, DI, and/or BP
Return: AX = return from function
Program: DeskMate is a proprietary GUI from Tandy distributed with several
    models of the Tandy 1000's, 2500's, 3000's, and laptops.  Retail
    and runtime versions also exist.  Some Tandy's are designed to
    boot directly into DeskMate.
SeeAlso: INT 15/AX=7002h,INT E1"DeskMate"
--------O-E0--------------------------------
INT E0 - CP/M-86, Concurrent CP/M, DR Multiuser DOS - FUNCTION CALLS
```

```
  CL = function number (see #04019,#04020)
  DS,DX contain parameter(s):
    DL = byte parameter
    DX = word parameter
    DS:DX -> structure
Return: as appropriate for function:
    AL = byte result
    AX = word
    ES:AX -> structure (and BX=ES)
  CX is often the error code (see #04021)
Notes:  several functions are covered in more detail in following entries
  most of these calls are also supported by Digital Research's DOS Plus
    v2.1; the unsupported functions are 26h,29h-2Bh,3Ah,3Dh-62h,71h-8Ch,
    90h-92h,94h-97h,9Bh-ABh, and AEh-FFh
SeeAlso: INT 21/AX=4459h,INT 21/AH=E0h"DOS Plus",INT E6"CP/M-86"

(Table 04019)
Values for CP/M-86,DR Multiuser DOS function number:
 00h  terminate calling process      (see INT E0/CL=00h)
 01h  read a character         (see INT E0/CL=01h)
 02h  write character to default console     (see INT E0/CL=02h)
 03h  read character from default AUX      (see INT E0/CL=03h)
 04h  write character to default AUX       (see INT E0/CL=04h)
 05h  write character to default list device     (see INT E0/CL=05h)
 06h  perform raw I/O on default console     (see INT E0/CL=06h)
 07h  return default AUX input status      (see INT E0/CL=07h)
 08h  return default AUX output status     (see INT E0/CL=08h)
 09h  write string to default console      (see INT E0/CL=09h)
 0Ah  read string from default console     (see INT E0/CL=0Ah)
 0Bh  return default console input status    (see INT E0/CL=0Bh)
 0Ch  get BDOS release ID        (see INT E0/CL=0Ch)
 0Dh  reset all disk drives    (see also INT 21/AH=0Dh)
 0Eh  set default drive    (see also INT 21/AH=0Eh"DOS 1+")
 0Fh  open file via FCB    (see also INT 21/AH=0Fh,#01345)
 10h  close file via FCB     (see also INT 21/AH=10h)
 11h  search for first matching file with FCB (see also INT 21/AH=11h)
 12h  search for next matching file with FCB (see also INT 21/AH=12h)
 13h  delete file via FCB    (see also INT 21/AH=13h)
 14h  sequential read via FCB    (see also INT 21/AH=14h)
 15h  sequential write via FCB   (see also INT 21/AH=15h)
 16h  create file via FCB    (see also INT 21/AH=16h)
```

```
17h  rename file via FCB   (see also INT 21/AH=17h)
18h  get bit map of logged drives
19h  get default drive   (see also INT 21/AH=19h)
1Ah  set DMA address offset
1Bh  get default disk allocation vector (see also INT 21/AH=1Bh)
1Ch  set default drive to read-only
1Dh  get bit map of read-only drives
1Eh  set file attributes via FCB (see also INT 21/AX=4301h)
1Fh  get address of disk parameter block (see also INT 21/AH=1Fh)
20h  get/set default user number
21h  read random record via FCB  (see also INT 21/AH=21h)
22h  write random record via FCB (see also INT 21/AH=22h)
23h  compute file size with FCB  (see also INT 21/AH=23h)
24h  get FCB random record number  (see also INT 21/AH=24h)
25h  reset specified drives
26h  access specified drives (not in DR DOS Plus v2.1)
27h  free specified drives
28h  write random with FCB, zero fill (see also INT 21/AH=28h)
2Ah  lock records in FCB file  (see also INT 21/AH=5Ch)
2Bh  unlock records in FCB file  (see also INT 21/AH=5Ch)
2Ch  set BDOS multisector count
2Dh  set BDOS error mode
2Eh  get free space on disk
2Fh  load, initialize, and jump to process ("chain process")
 (see INT E0/CL=2Fh,INT 21/AH=4Bh)
30h  flush write-deferred buffers
31h  get/set system variable (DOS Plus v2.1)
32h  call BIOS (XIOS) character routine (see #04020)
33h  set DMA address segment
34h  get DMA buffer address
35h  CP/M-86 allocate maximum memory     (see INT E0/CL=35h)
36h  allocate maximum memory at specified segment  (see INT E0/CL=36h)
37h  CP/M-86 allocate memory segment (see INT E0/CL=37h,INT 21/AH=48h)
38h  allocate memory at specified segment    (see INT E0/CL=38h)
39h  CP/M-86 free specified memory segment (see INT E0/CL=39h,INT 21/AH=49h)
3Ah  CP/M-86 free all memory (not in DOS Plus v2.1)  (see INT E0/CL=3Ah)
3Bh  load .CMD file into memory      (see INT E0/CL=3Bh)
3Ch  (DOS Plus v2.1) call RSX program
40h  (DR-NET, REAL/32) log on a server   (see INT E0/CL=40h)
41h  (DR-NET, REAL/32) log off a server    (see INT E0/CL=41h)
42h  (DR-NET) send a message
```

```
43h  (DR-NET) receive a message
44h  (DR-NET, REAL/32) get network status    (see INT E0/CL=44h)
45h  (DR-NET, REAL/32) get requestor config table  (see INT E0/CL=45h)
46h  (DR-NET) set compatibility attributes
47h  (DR-NET, REAL/32) get server configuration table (see INT E0/CL=47h)
48h  (DR-NET, REAL/32) set network error mode  (see INT E0/CL=48h)
49h  (DR-NET, REAL/32) attach network
4Ah  (DR-NET, REAL/32) detach network
4Bh  (DR-NET, REAL/32) set default password
4Ch  (DR-NET, REAL/32) get-set long timeout
4Dh  (DR-NET, REAL/32) get parameter table
4Fh  (REAL/32) get extended network error
50h  (DR-NET, REAL/32) get network information
53h  get current time      (see also INT 21/AH=2Ch)
54h  set current time      (see also INT 21/AH=2Dh)
55h  get binary system date      (see also INT 21/AH=2Ah)
56h  set system date       (see also INT 21/AH=2Bh"DATE")
57h  allocate system flag
58h  deallocate system flag
59h  reserve memory in global area     (see INT E0/CL=59h)
5Ah  lock physical drive
5Bh  unlock physical drive
5Ch  search path for executable file     (see INT E0/CL=5Ch)
5Dh  load and execute command      (see INT E0/CL=5Dh)
5Eh  get/set process exit code     (see INT E0/CL=5Eh)
5Fh  set country information
60h  get country information
63h  truncate FCB file (see also INT 21/AH=28h)
64h  create/update directory label
65h  get directory label
66h  get FCB date stamp and password mode
67h  write extended FCB
68h  set system date and time
69h  get system date and time in binary
6Ah  establish password for file access
6Bh  get OS serial number        (see INT E0/CL=6Bh)
6Ch  (DOS Plus v2.1) get/set program return code
6Dh  get/set console mode        (see INT E0/CL=6Dh)
6Eh  get/set string delimiter      (see INT E0/CL=6Eh)
6Fh  write block to default console      (see INT E0/CL=6Fh)
70h  write block to default list device    (see INT E0/CL=70h)
```

71h execute DOS-compatible function (see INT E0/CL=71h)
74h set FCB time and date stamps
80h allocate memory (see INT E0/CL=80h)
82h deallocate memory (see INT E0/CL=81h)
83h poll I/O device
84h wait on system flag (see INT E0/CL=84h)
85h set system flag (see INT E0/CL=85h)
86h create message queue (see INT E0/CL=86h)
87h open message queue (see INT E0/CL=87h)
88h delete message queue (see INT E0/CL=88h)
89h read from message queue (see INT E0/CL=89h)
8Ah conditionally read from message queue (see INT E0/CL=8Ah)
8Bh write to message queue (see INT E0/CL=8Bh)
8Ch conditionally write to message queue (see INT E0/CL=8Ch)
8Dh delay calling process (see INT E0/CL=8Dh)
8Eh call process dispatcher (yield CPU) (see INT E0/CL=8Eh)
8Fh terminate calling process (same as func 00h) (see INT E0/CL=8Fh)
90h create a process (see INT E0/CL=90h)
91h set calling process' priority (see INT E0/CL=91h)
92h attach to default console (see INT E0/CL=92h)
93h detach from default console (see INT E0/CL=93h)
94h (REAL/32) set the process' default console (see INT E0/CL=94h)
95h assign default console to process (see INT E0/CL=95h)
96h interpret and execute commandline (see INT E0/CL=96h)
97h resident procedure library
98h parse ASCII string into FCB (see also INT 21/AH=29h)
99h return default console (see INT E0/CL=99h)
9Ah get address of system data (SYSDAT) (see INT E0/CL=9Ah)
9Bh get system time and date
9Ch return calling process' descriptor (see INT E0/CL=9Ch)
9Dh terminate process by name or PD address (see INT E0/CL=9Dh)
9Eh attach to default list device (see INT E0/CL=9Eh)
9Fh detach from default list device (see INT E0/CL=9Fh)
A0h select default list device (see INT E0/CL=A0h)
A1h conditionally attach to default list device (see INT E0/CL=A1h)
A2h conditionally attach to default console (see INT E0/CL=A2h)
A3h get OS version number (see INT E0/CL=A3h)
A4h get default list device (see INT E0/CL=A4h)
A5h attach to default AUX (see INT E0/CL=A5h)
A6h detach from default AUX (see INT E0/CL=A6h)
A7h conditionally attach to default AUX (see INT E0/CL=A7h)

```
 A8h  set default AUX          (see INT E0/CL=A8h)
 A9h  return default AUX         (see INT E0/CL=A9h)
 ACh  read block from default AUX      (see INT E0/CL=ACh)
 ADh  (DOS Plus v2.1) write block to default AUX  (see INT E0/CL=ADh)
 B0h  configure default AUX        (see INT E0/CL=B0h)
 B1h  get/set device control parameters    (see INT E0/CL=B1h)
 B2h  send Break through default AUX       (see INT E0/CL=B2h)
 B3h  allocate physical memory       (see INT E0/CL=B3h)
 B4h  free physical memory         (see INT E0/CL=B4h)
 B5h  map physical memory        (see INT E0/CL=B5h)
 B6h  nondestructive conditional message queue read (see INT E0/CL=B6h)
 B7h  timed wait on system flag      (see INT E0/CL=B7h)
 B8h  get/set I/O port mapping       (see INT E0/CL=B8h)
 B9h  set list device timeout        (see INT E0/CL=B9h)
 BAh  set AUX timeout value        (see INT E0/CL=BAh)
 BBh  execute XIOS service
 BDh  (DR Multiuser DOS) delay       (see INT E0/CL=BDh)
 FFh  return 80386 to native mode
SeeAlso: #04020,#04021

(Table 04020)
Values for DOS Plus v2.1 XIOS functions:
 00h  terminate program
 01h  ???
 02h  check for console input status
 03h  read character from console
 04h  write character to console
 05h  write character to list device
 06h  write character to auxiliary device
 07h  read character from auxiliary device
 0Fh  get list device status
 10h-14h reserved
 15h  device initialization
 16h  check console output status
 17h-7Fh reserved
---BBC Acorn---
 80h  get XIOS version
 81h  get Tube semaphore
 82h  release Tube semaphore
 83h  select text/graphics
 84h  update B&W graphics rectangle
```

```
 85h  update color graphics rectangle
 86h  get/release/update mouse
 87h  get system error info
 88h  entry in CLOCK called by WatchDog RSP
 89h  BBC OSBYTE function
 8Ah  BBC OSWORD function
SeeAlso: #04019

(Table 04021)
Values for DR Multiuser DOS Error Return Code:
 00h  no error
 01h  system call not implemented
 02h  illegal system call number
 03h  cannot find memory
 04h  illegal flag number
 05h  flag overrun
 06h  flag underrun
 07h  no unused Queue Descriptors
 08h  no free queue buffer
 09h  cannot find queue
 0Ah  queue in use
 0Ch  no free Process Descriptors
 0Dh  no queue access
 0Eh  empty queue
 0Fh  full queue
 10h  CLI queue missing
 11h  no 8087 in system
 12h  no unused Memory Descriptors
 13h  illegal console number
 14h  no Process Descriptor match
 15h  no console match
 16h  no CLI process
 17h  illegal disk number
 18h  illegal filename
 19h  illegal filetype
 1Ah  character not ready
 1Bh  illegal Memory Descriptor
 1Ch  bad return from BDOS load
 1Dh  bad return from BDOS read
 1Eh  bad return from BDOS open
 1Fh  null command
```

```
 20h  not owner of resource
 21h  no CSEG in load file
 22h  process Descriptor exists on Thread Root
 23h  could not terminate process
 24h  cannot attach to process
 25h  illegal list device number
 26h  illegal password
 28h  external termination occurred
 29h  fixup error upon load
 2Ah  flag set ignored
 2Bh  illegal auxilliary device number
SeeAlso: #04019
--------O-E0----CL00-------------------------
INT E0 - REAL/32 - "P_TERMCPM" - TERMINATE CALLING PROCESS
  CL = 00h
Return: AX = error code
      FFFFh on failure
  may destroy SI,DI???
Program: REAL/32 is the descendant of IMS Multiuser DOS, which in turn is
    derived from DR Multiuser DOS and its predecessors (Concurrent DOS,
    etc.)
Notes:  sets the exit code (ERRORLEVEL) to 00h
  INT E0h is officially reserved for Digital Research by Intel.
  Apparently some Borland products also used this interrupt, which forced
    Digital Research to indirect calls through other interrupt entry
    points.
  This is also supported by IMS Multiuser DOS and IMS REAL/32.
SeeAlso: INT 21/AH=00h
--------O-E0----CL01-------------------------
INT E0 - REAL/32 - "C_READ" - FETCH CHARACTER FROM THE DEFAULT CONSOLE
  CL = 01h
Return: AX = character
  may destroy SI,DI???
Notes:  this function echos the character to the screen, expanding Tab
    characters to the next multiple of eight columns; Ctrl-C is
    ignored if the calling process can not terminate
  the calling process is suspended until a character is available; if
    the caller does not own the console, it is suspended until it
    can attach to the console
SeeAlso: INT E0/CL=02h,INT E0/CL=06h,INT 21/AH=01h
--------O-E0----CL02-------------------------
```

```
INT E0 - REAL/32 - "C_WRITE" - WRITE CHARACTER TO DEFAULT CONSOLE
  CL = 02h
  DX = character
Return: nothing
  may destroy SI,DI???
Note: Tab characters are expanded to blanks up to the next multiple of
    eight columns
SeeAlso: INT E0/CL=01h,INT E0/CL=06h,INT 21/AH=02h
--------O-E0----CL03-------------------------
INT E0 - DR Multiuser DOS - "A_READ" - READ CHARACTER FROM DEFAULT AUX DEVICE
  CL = 03h
Return: AL = ASCII character
  may destroy SI,DI???
Notes:  A_READ reads the next 8-bit character from the logical auxilliary
    input device (AUXn:); control is not returned to the calling
    process until a character has been read.
  if another process owns AUX, this call blocks until the device becomes
    available
  this function is also supported by REAL/32
SeeAlso: INT 21/AH=03h,INT E0/CL=04h,INT E0/CL=07h,INT E0/CL=A5h,INT E0/CL=ACh
--------O-E0----CL04-------------------------
INT E0 - DR Multiuser DOS - "A_WRITE" - WRITE CHARACTER TO DEFAULT AUX DEVICE
  CL = 04h
  DL = BYTE to write
Return: nothing
  may destroy SI,DI,DH???
Note: if another process owns AUX, this call blocks until the device becomes
    available
SeeAlso: INT 21/AH=04h,INT E0/CL=03h,INT E0/CL=08h,INT E0/CL=A5h,INT E0/CL=ADh
--------O-E0----CL05-------------------------
INT E0 - REAL/32 - "L_WRITE" - WRITE CHARACTER TO DEFAULT LIST DEVICE
  CL = 05h
  DL = char to write
Return: nothing
  may destroy SI,DI???
Note: if another process owns the list device, this call blocks until the
    device becomes available
SeeAlso: INT 21/AH=05h
--------O-E0----CL06-------------------------
INT E0 - REAL/32 - "C_RAWIO" - PERFORM RAW I/O WITH DEFAULT CONSOLE
  CL = 06h
```

```
  DL = mode describing the operation to be performed
       FFh get console input/status
       FEh get console status
       FDh get console input (blocking)
       else output DL to the console as a character
Return: AX = returned value
       for DL = FFh, the character or 00h if none available
       for DL = FEh, 00h if no characters available, FFh if any available
       for DL = FDh, the character read from the console
       else AX = 0000h
  may destroy SI,DI???
Notes:  during raw I/O, the special characters ^C, ^O, ^P, and ^S are not
    interpreted, but are passed through
  if the virtual console is in ^S mode and the owning process calls
    this function, the ^S state is cleared
SeeAlso: INT E0/CL=01h,INT E0/CL=02h,INT 21/AH=06h
--------O-E0----CL07-------------------------
INT E0 - DR Multiuser DOS - "A_STATIN" - GET INPUT STATUS OF AUX DEVICE
  CL = 07h
Return: AL = status
       00h not ready
       FFh character available
Desc: determine whether the current AUX device has input available
SeeAlso: INT E0/CL=03h,INT E0/CL=08h
--------O-E0----CL08-------------------------
INT E0 - DR Multiuser DOS - "A_STATOUT" - GET OUTPUT STATUS OF AUX DEVICE
  CL = 08h
Return: AL = status
       00h not ready
       FFh ready for output
Desc: determine whether the current AUX device is able to accept more output
SeeAlso: INT E0/CL=04h,INT E0/CL=07h
--------O-E0----CL09-------------------------
INT E0 - REAL/32 - "C_WRITESTR" - WRITE STRING TO DEFAULT CONSOLE
  CL = 09h
  DS:DX -> string
Return: nothing
  may destroy SI,DI,DS???
Note: the string terminated with a '$' character (24h) by default; the
    terminator may be changed with C_DELIMIT
  tabs are expanded to the next multiple of eight columns
```

```
SeeAlso: INT E0/CL=6Eh
--------O-E0----CL0A-------------------------
INT E0 - REAL/32 - "C_READSTR" - READ STRING FROM DEFAULT CONSOLE
  CL = 0Ah
  DS:DX -> buffer for string (see #04022)
Return: nothing

Format of REAL/32 "C_READSTR" buffer:
Offset  Size  Description (Table 04022)
 00h  BYTE  maximum number of characters buffer can hold
 01h  BYTE  actual number of buffers read
 02h  N BYTEs input line
--------O-E0----CL0B-------------------------
INT E0 - REAL/32 - "C_STAT" - RETURN DEFAULT CONSOLE INPUT STATUS
  CL = 0Bh
Return: AX = status
      0000h no characters ready
      0001h character available
  may destroy SI,DI???
Note: after setting bit 0 of the console mode word with C_MODE, this function
    will only return AX=0001h when the user presses Ctrl-C.
--------O-E0----CL0C-------------------------
INT E0 - REAL/32 - "S_BDOSVER" - GET BDOS VERSION
  CL = 0Ch
Return: AX = version (see #04023)
  may destroy SI,DI???
SeeAlso: INT E0/CL=A3h,INT 21/AX=4451h

(Table 04023)
Values for REAL/32 BDOS version:
 1432h  - DR Concurrent PC DOS Version 3.2
 1441h  - DR Concurrent DOS Version 4.1
 1450h  - DR Concurrent DOS/XM Version 5.0
 1463h  - DR Multiuser DOS Release 5.0
 1465h  - DR Multiuser DOS Release 5.01
 1466h  - DR Multiuser DOS Release 5.1, IMS Multiuser DOS Enhanced Release 5.1
 1467h  - IMS Multiuser DOS Version 7.0, 7.1
 1468h  - IMS REAL/32 Version 7.50, 7.51
 1469h  - IMS REAL/32 Version 7.52, 7.53
 14??h  - IMS REAL/32 Version 7.6
SeeAlso: #01579,#01580,#01581
```

```
--------O-E0----CL2F-------------------------
INT E0 - REAL/32 - "P_CHAIN" - CHAIN PROCESS
  CL = 2Fh
  [DTA] = ASCIZ command line for process to start
Return: AX = return code
      0000h successful
      FFFFh failed
  may destroy SI,DI???
--------O-E0----CL35-------------------------
INT E0 R - REAL/32 - "MC_MAX" - CP-M/86 ALLOCATE MAXIMUM MEMORY
  CL = 35h
  DS:DX -> MCB (see #04024)
Return: AX = status
  may destroy SI,DI,DS???
SeeAlso: INT E0/CL=39h

Format of REAL/32 MCB (Memory Control Block):
Offset  Size  Description (Table 04024)
 00h  WORD  segment address of memory block
 02h  WORD  length of block in paragraphs
 04h  BYTE  reserved (0)
--------O-E0----CL36-------------------------
INT E0 R - REAL/32 - "MC_ABSMAX" - ALLOCATE MAXIMUM MEMORY SEGMENT ABSOLUTE
  CL = 36h
  DS:DX -> MCB (see #04024)
Return: AX = status
  may destroy SI,DI,DS???
SeeAlso: INT E0/CL=39h
--------O-E0----CL37-------------------------
INT E0 R - REAL/32 - "MC_ALLOC" - CP-M/86 ALLOCATE MEMORY SEGMENT
  CL = 37h
  DS:DX -> MCB (see #04024)
Return: AX = status
  may destroy SI,DI,DS???
SeeAlso: INT E0/CL=39h
--------O-E0----CL38-------------------------
INT E0 R - REAL/32 - "MC_ABSALLOC" - ALLOCATE MEMORY SEGMENT ABSOLUTE
  CL = 38h
  DS:DX -> MCB (see #04024)
Return: AX = status
  may destroy SI,DI,DS???
```

```
SeeAlso: INT E0/CL=39h
--------O-E0----CL39-------------------------
INT E0 R - REAL/32 - "MC_FREE" - CP-M/86 FREE SPECIFIED MEMORY SEGMENT
  CL = 39h
  DS:DX -> MCB (see #04024)
Return: AX = status
  may destroy SI,DI,DS???
SeeAlso: INT E0/CL=3Ah
--------O-E0----CL3A-------------------------
INT E0 R - REAL/32 - "MC_ALLFREE" - CP-M/86 FREE ALL MEMORY
  CL = 3Ah
Return: nothing???
Desc: release all of the calling process's memory except the User Data Area
SeeAlso: INT E0/CL=39h
--------O-E0----CL3B-------------------------
INT E0 u - REAL/32 - "P_LOAD" - LOAD .CMD FILE INTO MEMORY
  CL = 3Bh
  ???
Return: ???
Note: IMS does not document the details of this call because .CMD files are
    supported for backward compatibility only
--------O-E0----CL40-------------------------
INT E0 - REAL/32 - "N_LOGON" - LOG ONTO A SERVER
  CL = 40h
  DS:DX -> LPB (see #04025)
Return: AX = status (0000h,00FFh,07FFh,0DFFh,0EFFh,FFFFh) (see #04030)
  may destroy SI,DI,DS???
SeeAlso: INT E0/CL=41h

Format of REAL/32 Logon/Logoff Parameter Block:
Offset  Size  Description (Table 04025)
 00h  BYTE  node ID
 01h  8 BYTEs password for server access
 09h  DWORD -> process descriptor of process to be logged on (see #04026)

Format of REAL/32 Process Descriptor:
Offset  Size  Description (Table 04026)
 00h  WORD  offset of next entry in current descriptor list
 02h  WORD  offset of thread list
 04h  BYTE  current processor status (see #04027)
 05h  BYTE  priority
```

```
 06h  WORD  runtime flags (see #04028)
 08h  8 BYTEs process name
 10h  WORD  segment of User Data Area (UDA) (256 or 352 bytes)
 12h  BYTE  current default disk drive
 13h  BYTE  drive from which process was loaded
 14h  WORD  reserved (0)
 16h  WORD  offset of memory descriptor list for memory owned by process
 18h  6 BYTEs reserved
 1Eh  WORD  offset of process descriptor for parent process
    0000h if parent has already terminated
 20h  BYTE  number of default console
 21h  BYTE  number of default AUX device
 22h  2 BYTEs reserved
 24h  BYTE  number of default list device
 25h  BYTE  reserved
 26h  WORD  flags (see #04029)
 28h 12 BYTEs reserved
 34h  WORD  offset of memory page allocation root
 36h 22 BYTEs reserved
Note: At least offset 10h (UDA) is also supported by MDOS 5.00, 5.01, 5.10,
    as the DR DOS MEM utility retrieves this UDA segment through
    INT E0/CL=9Ah when INT 21/AX=4451h returns 1463h, 1465h, or 1466h.
SeeAlso: #04025

(Table 04027)
Values for REAL/32 processor status:
 00h  process is ready to run
 01h  process is polling a device
 02h  delaying for a number of ticks
 03h  in swap list
 04h  terminating
 05h  asleep
 06h  waiting to read a message from a queue
 07h  waiting to write a message
 08h  waiting on system flag or semaphore
 09h  waiting to attach to an I/O device
 0Ah  waiting on sync block
 0Bh  waiting for system flag or semaphore with timeout
 0Ch  forced dispatch
SeeAlso: #04026
```

```
Bitfields for REAL/32 process flags:
Bit(s)  Description (Table 04028)
 0  system process
 1  do not terminate
 2  special rsp process
 3  from process descriptor table
 14-4 unused???
 15 uses math coprocessor
SeeAlso: #04026,#04029

Bitfields for REAL/32 secondary flags:
Bit(s)  Description (Table 04029)
 0  suspend when in background
 1  ???
 2  in foreground
 3  enable APPEND processing on file opens
 4  Ctrl-C was typed
 5  Ctrl-C will reset disk system
 6  OK to read locked record
 7  ???
 8  do not perform banking
SeeAlso: #04026,#04028

(Table 04030)
Values for REAL/32 "N_LOGON" status:
 0000h  successful
 00FFh  server could not create shadow process
 07FFh  incorrect password
 0CFFh  not logged into specified server
 0DFFh  process already logged onto 16 servers,
  LPB process not attached to network
 0EFFh  physical transmission prevented message or response from getting thru
  network error during logoff
 FFFFh  calling process not attached to network
--------O-E0----CL41-------------------------
INT E0 - REAL/32 - "N_LOGON" - LOG OFF A SERVER
  CL = 41h
  DS:DX -> LPB (see #04025)
Return: AX = status (0000h,0CFFh,0DFFh,0EFFh,FFFFh) (see #04030)
  may destroy SI,DI,DS???
SeeAlso: INT E0/CL=40h
```

```
--------O-E0----CL44-------------------------
INT E0 - REAL/32 - "N_STAT" - GET NETWORK STATUS
  CL = 44h
Return: AX = network status or 0FFFh on error
      bit 8: calling process is attached to network
  may destroy SI,DI???
--------O-E0----CL45-------------------------
INT E0 - REAL/32 - "N_RCT" - GET REQUESTOR CONFIGURATION TABLE
  CL = 45h
  DS:DX -> RCT Control Block (see #04031,#04032)
Return: AX = status or error code (see #04033,#04030)
  may destroy SI,DI,DS???
SeeAlso: INT E0/CL=47h

Format of REAL/32 RCT Control Block for Drives:
Offset  Size  Description (Table 04031)
 00h  BYTE  command byte
    00h map disk drive
    02h map list device
 01h  BYTE  local resource ID
 02h  BYTE  remote resource ID
 03h  BYTE  server node ID number
Note: the RCT Control Block is a union of two possible structures; this
    variant is used when mapping disk drives and list devices
SeeAlso: #04032

Format of REAL/32 RCT Control Block for Queues:
Offset  Size  Description (Table 04032)
 00h  BYTE  command byte
    03h map local queue to server
    04h unmap queue
 01h  8 BYTEs local queue name (case-sensitive)
 09h  8 BYTEs remote queue name (case-sensitive)
 11h  BYTE  server node ID number
Note: the RCT Control Block is a union of two possible structures; this
    variant is used when mapping queues
SeeAlso: #04031

(Table 04033)
Values for REAL/32 "N_RCT" status:
 0000h  successful
```

```
 0001h  invalid local device
 0002h  invalid remote device
 0003h  no queue entry space
--------O-E0----CL47-------------------------
INT E0 - REAL/32 - "N_SCT" - GET SERVER CONFIGURATION TABLE
  CL = 47h
  DS:DX -> 112-byte buffer for Server Configuration Table (see #04034)
Return: AX = status (0000h successful, else error code)
  may destroy SI,DI,DS???
Note: the first byte of the SCT buffer is set to the desired server number
    prior to calling this function
SeeAlso: INT E0/CL=45h

Format of REAL/32 Server Configuration Table:
Offset  Size  Description (Table 04034)
 00h  BYTE  server's default physical drive ID
 01h  BYTE  network status
 02h  BYTE  server node ID number
 03h  BYTE  maximum number of requesters supported by server
 04h  BYTE  current number of shadow processes
 05h 108 BYTEs  six logon structures, each:
    Offset  Size  Description
     00h  WORD  bit vector of logged-in nodes
     02h 16 BYTEs list of requester node IDs
--------O-E0----CL48-------------------------
INT E0 - REAL/32 - "N_ERRMODE" - SET NETWORK ERROR MODE
  CL = 48h
  DL = new error mode
       FFh return error in registers AX,BX,CX
       FEh display message and return error
       FDh display message and abort (default)
Return: CX = error code (0000h successful, FFFFh failed)
  may destroy SI,DI???
Desc: specify how the REAL/32 Net Server responds to error numbers 0CFFh,
    0DFFh, and 0EFFh (see #04030)
--------O-E0----CL59-------------------------
INT E0 - ConcCP/M,DR Multiuser DOS - "S_MEMORY" - RESERVE MEMORY IN GLOBAL AREA
  CL = 59h
  DX = size in bytes
Return: AX = status
       FFFFh failed
```

```
      other successful
    ES:BX -> reserved memory
--------O-E0----CL5C-------------------------
INT E0 - REAL/32 - "P_PATH" - SCAN PATH FOR EXECUTABLE FILE
  CL = 5Ch
  DS:DX -> Exec Parameter Block (EPB) (see #04035)
Return: AX = status
      FFFEh file not found
      FFFDh path not found
      FFFCh too many handles open
      FFFBh access denied
      FFF6h invalid environment
      FFDCh sharing conflict
      FFAAh invalid password
  EPB updated (if successful)
  may destroy SI,DI,DS
SeeAlso: INT E0/CL=5Dh,INT E0/CL=5Eh,INT 21/AH=4Bh

Format of REAL/32 Exec Parameter Block (EPB):
Offset  Size  Description (Table 04035)
 00h  DWORD -> ASCIZ command to be executed
 04h  BYTE  type of exec
    00h .CMD
    01h .COM
    02h .EXE
    03h .BAT
    04h RSP-type queue
 05h  BYTE  flags
    bit 0: overlay existing program
    bit 1: don't assign console to child
    bit 2: allocate memory for .CMD within parent's memory space
    bit 3: make full banked window available while loading child
 06h  DWORD 32-bit offset to ASCIZ command line
 0Ah  WORD  selector for ASCIZ command line
--------O-E0----CL5D-------------------------
INT E0 - REAL/32 - "P_EXEC" - EXECUTE CHILD PROCESS
  CL = 5Dh
  DS:DX -> Exec Parameter Block (EPB) (see #04035)
Return: AX = status
      FFFEh file not found
      FFFDh path not found
```

```
      FFFCh too many handles open
      FFFBh access denied
      FFF6h invalid environment
      FFDCh sharing conflict
      FFAAh invalid password
  EPB updated (if successful)
  may destroy SI,DI,DS
SeeAlso: INT E0/CL=5Ch,INT E0/CL=5Eh,INT 21/AH=4Bh
--------O-E0----CL5E-------------------------
INT E0 - REAL/32 - "P_EXITCODE" - GET/SET PROCESS EXIT CODE
  CL = 5Eh
  DX = exit code or FFFFh to get
       DH ignored when setting code
       DL = process exit code
Return: AX = status (FFFFh on error)
  if getting:
       AH = termination reason
    00h normal
    01h external termination via Ctrl-C or P_ABORT
    02h hardware (critical) error abort
    03h child did not terminate
    FFh illegal request (no child, or exit code already retrieved)
      AL = process exit code
SeeAlso: INT E0/CL=5Dh,INT 21/AH=4Ch
--------O-E0----CL6B-------------------------
INT E0 - REAL/32 - "S_SERIAL" - GET OS SERIAL NUMBER
  CL = 6Bh
  DS:DX -> 6-byte buffer for ASCII serial number
Return: nothing???
--------O-E0----CL6D-------------------------
INT E0 - REAL/32 - "C_MODE" - GET/SET CONSOLE MODE
  CL = 6Dh
  DX = new console mode (see #04036), or FFFFh to get current mode
Return: AX = status if setting (0000h = success)
  AX = current console mode if DX=FFFFh on entry

Bitfields for REAL/32 console mode:
Bit(s)  Description (Table 04036)
 0  C_STAT function will return 01h only when Ctrl-C is pressed
 1  disable support for stop/start scroll (Ctrl-S/Ctrl-Q)
 2  raw console output (no tab expansion, no printer echo)
```

```
 3  disable program termination on Ctrl-C
 7  disable Ctrl-O console bit-bucket
 10 enable Esc as end-of-line character
--------O-E0----CL6E-------------------------
INT E0 - REAL/32 - "C_DELIMIT" - GET/SET STRING DELIMITER
  CL = 6Eh
  DX = new delimiter, or FFFFh to get current string delimiter
Return: AX = status (0000h success) if setting
  AX = current string delimiter (default is 24h '$' at process start)
SeeAlso: INT E0/CL=09h
--------O-E0----CL6F-------------------------
INT E0 - REAL/32 - "C_WRITEBLK" - WRITE BLOCK TO DEFAULT CONSOLE
  CL = 6Fh
  DS:DX -> Character Control Block (see #04051,#04052)
Return: nothing???
SeeAlso: INT E0/CL=70h
--------O-E0----CL70-------------------------
INT E0 - REAL/32 - "L_WRITEBLK" - WRITE BLOCK TO DEFAULT LIST (PRINTER) DEVICE
  CL = 70h
  DS:DX -> Character Control Block (see #04051,#04052)
Return: nothing???
SeeAlso: INT E0/CL=6Fh
--------O-E0----CL71-------------------------
INT E0 R - ConcCP/M,DR Multiuser DOS - EXECUTE DOS-COMPATIBLE FUNCTIONS
  CL = 71h
  DS:DX -> parameter block (see #04038)
Return: AX = status (0000h successful, else error code)
  may destroy SI,DI,DS

(Table 04037)
Values for DR "execute DOS-compatible function" function number:
 00h  "fd_getdpb" get disk information (see also INT 21/AH=32h)
 01h  "fd_mkdir"  create directory (see also INT 21/AH=39h)
 02h  "fd_rmdir"  remove directory (see also INT 21/AH=3Ah)
 03h  "fd_chdir"  change directory (see also INT 21/AH=3Bh)
 04h  "fd_creat"  create file (see also INT 21/AH=3Ch)
 05h  "fd_open" open existing file (see also INT 21/AH=3Dh)
 06h  "fd_close"  close file (see also INT 21/AH=3Eh)
 07h  "fd_read" read from file (see also INT 21/AH=3Fh)
 08h  "fd_write"  write to file (see also INT 21/AH=40h)
 09h  "fd_delete" delete file (see also INT 21/AH=41h)
```

```
 0Ah  "fd_lseek"  get/set file position (see also INT 21/AH=42h)
 0Bh  "fd_chmod"  get/set file attributes (see also INT 21/AH=43h)
 0Ch  "fd_curdir" get current directory (see also INT 21/AH=47h)
 0Dh  "fd_sfirst" find first matching file (see also INT 21/AH=4Eh)
 0Eh  "fd_snext"  find next matching file (see also INT 21/AH=4Fh)
 0Fh  "fd_rename" rename file (see also INT 21/AH=56h)
 10h  "fd_gsfdate"  get/set file date (see also INT 21/AX=5700h)
 11h  "fd_mktemp" make temporary file (see also INT 21/AH=5Ah)
 12h  "fd_mknew"  create new file (see also INT 21/AH=5Bh)
 13h  "fd_lock" lock/unlock file data (see also INT 21/AH=5Ch)
 14h  "fd_dup"  duplicate file handle (see also INT 21/AH=45h)
 15h  "fd_dup2" force duplicate file handle (see also INT 21/AH=46h)
 16h-19h ???
 1Ah  "fd_ioctl"  I/O control emulation
 1Bh  "fd_commit" commit file to disk (see also INT 21/AH=68h)
 1Ch  "fd_expand" expand file name (see also INT 21/AH=60h)
 1Dh  ???
 1Eh  ???
 1Fh  "fd_sethandles" set number of handles for calling process
SeeAlso: #04038

Format of DR "execute DOS-compatible function" parameter block:
Offset  Size  Description (Table 04038)
 00h  WORD  function number (see #04037)
---function 00h---
 02h  WORD  drive
 04h  DWORD -> DPB (see #04039)
---function 01h,02h,03h---
 02h  DWORD -> ASCIZ directory name
---function 04h---
 02h  DWORD -> ASCIZ filename
 06h  WORD  file attributes
---function 05h---
 02h  DWORD -> ASCIZ filename
 06h  WORD  open mode (see INT 21/AH=3Dh)
---function 06h---
 02h  WORD  file handle
---function 07h,08h---
 02h  WORD  file handle
 04h  DWORD -> buffer
 08h  WORD  number of bytes to read or write
```

```
---function 09h---
 02h  DWORD -> ASCIZ filename
---function 0Ah---
 02h  WORD  file handle
 04h  DWORD (call) desired offset
    (ret) new file position if function is successful
 08h  WORD  seek origin (offset is calculated from specified position)
    00h start of file
    01h current position
    02h end of file
---function 0Bh---
 02h  DWORD -> ASCIZ filename
 06h  WORD  (call) new file attributes or new file owner
    (ret) current/new file attributes, depending on function
 08h  WORD  subfunction
    00h get attribute
    01h set attribute
    02h get extended attributes
    03h set extended attributes (and password)
    04h get encrypted password
    05h set extended attributes (and encrypted password)
    06h get file owner
    07h set file owner
    Note: the password is taken from the first 8 bytes of the DTA
---function 0Ch---
 02h  WORD  drive
 04h  DWORD -> 64-byte buffer for ASCIZ current directory path
---function 0Dh---
 02h  DWORD -> ASCIZ filespec
 06h  WORD  file attributes
 08h  WORD  (call) size of buffer pointed at by current DTA
    (ret) number of matching files returned???
    Note: if the volume label attribute is specified, the root
        directory will be searched
      if the buffer is at least 47 bytes, multiple matching
        entries will be stored in the DTA (at 21 bytes per
        entry)
---function 0Eh---
 02h  WORD  number of additional matches to store in DTA
    (normally set to 0, but if DTA is at least 47 bytes, can be
      set to (DTAsize-21)/26 to return multiple matches)
```

```
---function 0Fh---
 02h  DWORD -> ASCIZ filename
 06h  DWORD -> ASCIZ new name
---function 10h---
 02h  WORD  file handle
 04h  WORD  subfunction
    00h get date and time
    01h set date and time
 06h  WORD  file date
 08h  WORD  file time
---function 11h---
 02h  DWORD -> ASCIZ pathname + 13 extra bytes for generated filename
    (if pathname does not end in backslash, one will be appended)
 06h  WORD  file attributes
---function 12h---
 02h  DWORD -> ASCIZ filename
 06h  WORD  file attributes
---function 13h---
 02h  WORD  file handle
 04h  DWORD offset of start of region
 08h  DWORD length of region to lock/unlock
 0Ch  WORD  subfunction (00h = lock, 01h = unlock)
---function 14h---
 02h  WORD  file handle to be duplicated
    (new file handle is returned as function return value)
---function 15h---
 02h  WORD  file handle to be duplicated
 04h  WORD  file handle which is to become the duplicate
---function 1Ah, form 1---
 02h  WORD  file handle
 04h  WORD  IOCTL function number (00h,01h,06h,07h,0Ah)
 06h  WORD  (ret) status
---function 1Ah, form 2---
 02h  WORD  drive number
 04h  WORD  IOCTL function number (08h,09h,0Eh,0Fh)
 06h  WORD  (ret) status
---function 1Ah, form 3---
 02h  WORD  file handle
 04h  WORD  IOCTL function number (02h,03h,54h)
 06h  DWORD -> buffer
 0Ah  WORD  (call) size of buffer in bytes
```

```
    (ret) size of returned data
---function 1Ah, form 4---
 02h  WORD  drive number
 04h  WORD  IOCTL function number (04h,05h)
 06h  DWORD -> buffer
 0Ah  WORD  (call) size of buffer in bytes
    (ret) size of returned data
---function 1Bh---
 02h  WORD  file handle of file to be committed to disk
---function 1Ch---
 02h  DWORD -> ASCIZ relative filename
 06h  DWORD -> buffer for absolute filename
---function 1Fh---
 02h  WORD  desired number of file handles for process

Format of DR Multiuser DOS Disk Parameter Block (DPB):
Offset  Size  Description (Table 04039)
 00h  BYTE  drive number (00h = A:)
 01h  BYTE  relative unit number
 02h  WORD  sector size in bytes
 04h  BYTE  sectors per cluster - 1
 05h  BYTE  shift count to convert clusters into sectors
 06h  WORD  number of reserved sectors at beginning of drive
 08h  BYTE  number of FATs
 09h  WORD  number of root directory entries
 0Bh  WORD  number of first sector containing user data
 0Dh  WORD  number of clusterse on disk
 0Fh  BYTE  number of sectors per FAT
 10h  WORD  sector number of first directory sector
 12h  4 BYTEs reserved
 16h  BYTE  media ID byte (see #01356)
 17h  BYTE  00h if disk accessed, FFh if not
 18h  6 BYTEs reserved
 1Eh  WORD  number of free clusters on drive
Note: this structure is a subset of the MS-DOS 3.x Drive Parameter Block
SeeAlso: #04038,#01357 at INT 21/AH=1Fh,#01395 at INT 21/AH=32h
--------O-E0----CL73-------------------------
INT E0 - GSX-86, GEM/1, GEM/2 - API
  CL = 73h
  CH = 04h
  DS:DX -> parameter block
```

```
--------O-E0----CL80-------------------------
INT E0 - REAL/32 - "M_ALLOC" - ALLOCATE MEMORY
  CL = 80h
  DS:DX -> Memory Parameter Block (MPB) (see #04040)
Return: AX = status (0000h success, else error code)
SeeAlso: INT E0/CL=81h

Format of REAL/32 Memory Parameter Block (MPB):
Offset  Size  Description (Table 04040)
 00h  WORD  (call) desired starting paragraph of block, or
       0000h for anywhere
     (ret) starting paragraph of allocated block
 02h  WORD  (call) minimum number of paragraphs required
     (ret) actual number of paragraphs allocated
 04h  WORD  (call) maximum number of paragraphs to allocate
     (ret) actual number of paragraphs allocated
 06h  WORD  process descriptor (see #04026) of memory's owner or 0000h
 08h  WORD  flags (normally set to 0000h on call)
SeeAlso: #04041
--------O-E0----CL81-------------------------
INT E0 - REAL/32 - "M_FREE" - DEALLOCATE MEMORY
  CL = 81h
  DS:DX -> Memory Free Parameter Block (MFPB) (see #04041)
Return: AX = status (0000h success, else error code)
SeeAlso: INT E0/CL=81h

Format of REAL/32 Memory Free Parameter Block (MFPB) :
Offset  Size  Description (Table 04041)
 00h  WORD  starting segment of block to be freed
 02h  WORD  reserved (0)
SeeAlso: #04040
--------O-E0----CL84-------------------------
INT E0 - REAL/32 - "DEV_WAITFLAG" - WAIT ON SYSTEM FLAG
  CL = 84h
  DX = system flag ID
Return: AX = status (0000h success, else error code)
  if successful, this function does not return until the system flag has
    been set by an interrupt handler (see INT E0/CL=85h); if the flag was
    already set, this call returns immediately
SeeAlso: INT E0/CL=85h,INT E0/CL=B7h
--------O-E0----CL85-------------------------
```

```
INT E0 - REAL/32 - "DEV_SETFLAG" - SET SYSTEM FLAG
  CL = 85h
  DX = system flag ID
Return: AX = status (0000h success, else error code)
Note: REAL/32 returns an error if the flag was already set, which indicates
    that a previous logical interrupt has not yet been serviced
SeeAlso: INT E0/CL=84h
--------O-E0----CL86-------------------------
INT E0 - ConcCP/M,DR Multiuser DOS - "Q_MAKE" - CREATE MESSAGE QUEUE
  CL = 86h
  DS:DX -> queue descriptor (see #04042)
Return: AX = status (0000h success, FFFFh failure)
  CX = error code (see #04021)
Note: also supported by REAL/32
SeeAlso: INT E0/CL=87h

Format of DR Multiuser DOS queue descriptor:
Offset  Size  Description (Table 04042)
 00h  2 WORDs internal use, initialize to zeros
 04h  WORD  queue flags (see #04043)
 06h  8 BYTEs queue name
 0Eh  WORD  length of message
 10h  WORD  number of messages
 12h  4 WORDs internal use, initialize to zeros
 1Ah  WORD  offset in system area of buffer for messages

Bitfields for REAL/32 queue flags:
Bit(s)  Description (Table 04043)
 0  mutual exclusion queue
 1  can not be deleted
 2  restricted to system processes
 3  RSP message queue
 4  reserved for internal use
 5  RPL address queue
 7-6  reserved for internal use
 15-8 reserved for future use
SeeAlso: #04042
--------O-E0----CL87-------------------------
INT E0 - ConcCP/M,DR Multiuser DOS - "Q_OPEN" - OPEN MESSAGE QUEUE
  CL = 87h
  DS:DX -> queue parameter block (QPB) (see #04044,#04045)
```

```
Return: AX = status (0000h success, FFFFh failure)
  CX = error code (see #04021)
Note: also supported by REAL/32
SeeAlso: INT E0/CL=86h,INT E0/CL=88h,INT E0/CL=89h

Format of DR Multiuser DOS queue parameter block:
Offset  Size  Description (Table 04044)
 00h  WORD  internal use, initialize to zero
 02h  WORD  queue ID (set by INT E0/CL=87h)
 04h  WORD  internal use, initialize to zero
 06h  WORD  offset of queue message buffer
    (REAL/32) if FFFFh, then full address of buffer is stored at
      offest 10h
 08h  8 BYTEs queue name
---REAL/32 only----
 10h  DWORD segment:offset of queue message buffer
SeeAlso: #04045

Format of protected-mode REAL/32 Queue Parameter Block (QPB):
Offset  Size  Description (Table 04045)
 00h  WORD  internal use
 02h  WORD  queue ID
 04h  WORD  internal use
 06h  DWORD 32-bit offset to buffer
 0Ah  WORD  selector for buffer
 0Ch  8 BYTEs queue name
SeeAlso: #04044
--------O-E0----CL88-------------------------
INT E0 - REAL/32 - "Q_DELETE" - DELETE SYSTEM QUEUE
  CL = 88h
  DS:DX -> queue parameter block (QPB) (see #04044,#04045)
Return: AX = status (0000h success, FFFFh failure)
SeeAlso: INT E0/CL=87h
--------O-E0----CL89-------------------------
INT E0 - ConcCP/M,DR Multiuser DOS - "Q_READ" - READ MESSAGE QUEUE
  CL = 89h
  DS:DX -> queue parameter block (QPB) (see #04044,#04045)
Return: AX = status (0000h success, FFFFh failure)
  CX = error code (see #04021)
Note: also supported by REAL/32
SeeAlso: INT E0/CL=87h,INT E0/CL=8Ah,INT E0/CL=8Bh,INT E0/CL=B6h
```

```
--------O-E0----CL8A-------------------------
INT E0 - ConcCP/M,DR Multiuser DOS - "Q_CREAD" - CONDITIONALLY READ MSG QUEUE
  CL = 8Ah
  DS:DX -> queue parameter block (QPB) (see #04044,#04045)
Return: AX = status (0000h success, FFFFh failure)
  CX = error code (see #04021)
Note: also supported by REAL/32
SeeAlso: INT E0/CL=87h,INT E0/CL=89h,INT E0/CL=8Ch,INT E0/CL=B6h
--------O-E0----CL8B-------------------------
INT E0 - ConcCP/M,DR Multiuser DOS - "Q_WRITE" - WRITE MESSAGE QUEUE
  CL = 8Bh
  DS:DX -> queue parameter block (QPB) (see #04044,#04045)
Return: AX = status (0000h success, FFFFh failure)
  CX = error code (see #04021)
Note: also supported by REAL/32
SeeAlso: INT E0/CL=89h,INT E0/CL=8Ch
--------O-E0----CL8C-------------------------
INT E0 - ConcCP/M,DR Multiuser DOS - "Q_CWRITE" - CONDITIONALLY WRITE MSG QUEUE
  CL = 8Ch
  DS:DX -> queue parameter block (QPB) (see #04044,#04045)
Return: AX = status (0000h success, FFFFh failure)
  CX = error code (see #04021)
Note: also supported by REAL/32
SeeAlso: INT E0/CL=8Ah,INT E0/CL=8Bh
--------O-E0----CL8E-------------------------
INT E0 - ConcCP/M,DR Multiuser DOS - "P_DISPATCH" - CALL DISPATCHER
  CL = 8Eh
  DX = FFFFh (optional) to force dispatch
Return: nothing
Desc: allow other processes of the same or higher priority to run if they
    are ready
Notes:  if DX=FFFFh, a dispatch is forced even if no other process is ready
  also supported by REAL/32
SeeAlso: INT E0/CL=91h,INT 15/AX=1000h,INT 2F/AX=1680h
--------O-E0----CL8F-------------------------
INT E0 - REAL/32 - "P_TERM" - TERMINATE CALLING PROCESS
  CL = 8Fh
  DX = termination code
Return: never if successful
  AX = FFFFh on failure
Note: this function can not terminate processes whose KEEP flag is set
```

```
  if the termination code is FFh, this function can terminate the process
    even if its SYSTEM flag is on; otherwise, only user processes can
    terminate themselves
SeeAlso: INT E0/CL=90h,INT 21/AH=4Ch
--------O-E0----CL90-------------------------
INT E0 - REAL/32 - "P_CREATE" - CREATE A PROCESS
  CL = 90h
  DS:DX -> process descriptor in calling process' system memory area
    (see #04026)
Return: AX = status
Notes:  this call can create more than one process if the specied process
    descriptor's link (next-process) field is nonzero
  all reserved and unused fields in the process descriptor should be
    filled with zeros; passing an invalid descriptor or pointer may
    crash the system because the descriptor is not checked by the OS
  the newly-created process(es) is always a native process, which can not
    make DOS calls
SeeAlso: INT E0/CL=8Fh
--------O-E0----CL91-------------------------
INT E0 - ConcCP/M,DR Multiuser DOS - "P_PRIORITY" - SET PROCESS PRIORITY
  CL = 91h
  DL = new priority (00h highest to FFh lowest)
Return: nothing
Notes:  sets priority of calling process; transient processes are initialized
    to priority C8h
  also supported by REAL/32
SeeAlso: INT E0/CL=8Eh
--------O-E0----CL92-------------------------
INT E0 - ConcCP/M,DR Multiuser DOS - "C_ATTACH" - ATTACH TO DEFAULT CONSOLE
  CL = 92h
Return: AX = status
Notes:  also supported by REAL/32
  if the console is currently owned by another process, this function
    waits until the console is available
SeeAlso: INT E0/CL=93h,INT E0/CL=94h,INT E0/CL=99h,INT E0/CL=9Eh,INT E0/CL=A2h
--------O-E0----CL93-------------------------
INT E0 - ConcCP/M,DR Multiuser DOS - "C_DETACH" - DETACH FROM DEFAULT CONSOLE
  CL = 93h
Return: AX = status
      0000h successfully detached
      FFFFh detach failed
```

```
Note: also supported by REAL/32
SeeAlso: INT E0/CL=92h,INT E0/CL=A6h
--------O-E0----CL94-------------------------
INT E0 - REAL/32 - "C_SET" - SET PROCESS'S DEFAULT CONSOLE
  CL = 94h
  DX = console ID
Return: AX = status (0000h success, else error code)
SeeAlso: INT E0/CL=92h,INT E0/CL=95h,INT E0/CL=99h
--------O-E0----CL95-------------------------
INT E0 - REAL/32 - "C_ASSIGN" - ASSIGN DEFAULT CONSOLE TO ANOTHER PROCESS
  CL = 95h
  DS:DX -> Assign Control Parameter Block (ACPB) (see #04046)
Return: AX = status
SeeAlso: INT E0/CL=94h,INT E0/CL=99h

Format of REAL/32 Assign Control Parameter Block (ACPB):
Offset  Size  Description (Table 04046)
 00h  BYTE  ID of console to assign
 01h  BYTE  flag: if FFh, new process must have CNS as console for this
      call to succeed
 02h  DWORD -> process descriptor (see #04026) or 00000000h
 06h  8 BYTEs name of process to search for if descriptor field above is zero
--------O-E0----CL96-------------------------
INT E0 - REAL/32 - "P_CLI" - RUN COMMAND LINE INTERFACE
  CL = 96h
  DS:DX -> Command Line Buffer (CLBUF) (see #04047)
Return: AX = status
Desc: execute the indicated command concurrently with the calling process;
    system queue commands, .BAT, .CMD, .COM, and .EXE files can be
    executed
Note: the calling process will lose its virtual console and must reattach it
    before attempting any I/O
SeeAlso: INT E0/CL=92h

Format of REAL/32 Command Line Buffer (CLBUF):
Offset  Size  Description (Table 04047)
 00h  BYTE  reserved (0)
 01h 128 BYTEs  ASCIZ command line
 81h  BYTE  (0)
--------O-E0----CL99-------------------------
INT E0 - REAL/32 - "C_GET" - GET DEFAULT CONSOLE
```

```
  CL = 99h
Return: AX = default console ID
SeeAlso: INT E0/CL=94h
--------O-E0----CL9A-------------------------
INT E0 - REAL/32 - "S_SYSDAT" - GET SYSTEM DATA AREA
  CL = 9Ah
Return: ES:AX -> system data area (see #04048)
  may destroy SI,DI

Format of REAL/32 system data area:
Offset  Size  Description (Table 04048)
 00h  DWORD address of supervisor entry point
 04h 36 BYTEs reserved
 28h  DWORD address of XIOS entry point
 2Ch  DWORD address of XIOS initialization point
 30h  8 BYTEs reserved
 38h  DWORD address of IRET dispatcher entry point
 3Ch  DWORD address of RETF dispatcher entry point
 40h  WORD  segment of operating system code
 42h  WORD  paragraph address of first Resident System Process (RSP)
 44h  WORD  paragraph after OS system area
 46h  BYTE  reserved
 47h  BYTE  number of system console devices
 48h  BYTE  number of system list (printer) devices
 49h  BYTE  number of Character Control blocks
 4Ah  BYTE  number of system flags
 4Bh  BYTE  current search disk
 4Ch  WORD  maximum memory per process
 4Eh  BYTE  reserved
 4Fh  BYTE  "dayfile" (flag, true if FFh)
 50h  BYTE  default disk for temporary files
 51h  BYTE  system ticks per second (typically 60)
 52h  WORD  offset of Locked Unused list
 54h  WORD  offset of CCB table
 56h  WORD  offset of system flag table
 58h  WORD  offset of root of Memory Descriptor Unused list
 5Ah  WORD  offset of Memory Free list
 5Ch  WORD  offset of Process Unused list
 5Eh  WORD  offset of Queue Unused list
 60h  4 WORDs (no longer used) QMAU
 68h  WORD  offset of root of Ready List
```

```
 6Ah  WORD  offset of root of Delay List
 6Ch  WORD  offset of Dispatcher Ready list
 6Eh  WORD  offset of root of Poll List
 70h  WORD  reserved
 72h  WORD  offset of root of Thread List
 74h  WORD  offset of root of Queue List
 76h  WORD  offset of Memory Allocation list
 78h  WORD  segment of version string
 7Ah  WORD  BDOS version number
 7Ch  WORD  OS version number
 7Eh  WORD  number of days since 1978/01/01
 80h  BYTE  current time: hour
 81h  BYTE  current time: minute
 82h  BYTE  current time: second
 83h  BYTE  number of XIOS consoles
 84h  BYTE  number of XIOS list (printer) devices
 85h  BYTE  total number of character devices
 86h  WORD  offset of LCB table
 88h  WORD  bitmap of open files
 8Ah  BYTE  maximum locked records per process
 8Bh  BYTE  maximum open files per process
 8Ch  WORD  offset of process descriptor for math coprocessor's owner
 8Eh  WORD  offset of Auxiliary Control Block
 90h  8 BYTEs reserved
 98h  WORD  offset of root Memory Window Descriptor
 9Ah  WORD  reserved
 9Ch  BYTE  number of Auxiliary Control Blocks
 9Dh  BYTE  default search disk
 9Eh  BYTE  reserved
 9Fh  BYTE  number of physical consoles
 A0h  DWORD address of 8087 interrupt handler
 A4h  DWORD address of default 8087 exception handler
 ...    reserved
C00h 82 BYTEs XIOS header structure (see #04049)

Format of REAL/32 XIOS header structure:
Offset  Size  Description (Table 04049)
 00h  3 BYTEs XIOS initialization entry point
 03h  3 BYTEs XIOS service entry point
 06h  WORD  segment address of system data
 08h  DWORD address of supervisor entry point
```

```
 0Ch  BYTE  set tick flag
 0Dh  BYTE  ticks per second
 0Eh  BYTE  global Door Open interrupt flag
 0Fh  BYTE  number of Auxiliary Control Blocks
 10h  BYTE  number of physical consoles
 11h  BYTE  number of virtual consoles
 12h  BYTE  number of logical consoles
 13h  BYTE  number of list control blocks
 14h  WORD  offset of CCB table
 16h  WORD  offset of LCB table
 18h 16 WORDs offsets into DPH for drives A-P
 38h  WORD  buffer size in paragraphs
 3Ah  WORD  offset of ACB table
 3Ch  WORD  used by OEM
 3Eh  WORD  offset of CCB pointer array
 40h  WORD  offset of LCB pointer array
 42h  WORD  offset of ACB pointer array
 44h  BYTE  'first' flag
 45h  BYTE  unused
 46h  WORD  offset of print device support table
 48h  WORD  offset of aux device support table
 4Ah  WORD  XIOS extra segment
 4Ch  WORD  segment of save end address for debug
 4Eh  WORD  segment of save start address for debug
 50h  BYTE  number of main virtual console
 51h  WORD  segment of XIOS code
SeeAlso: #04048
--------O-E0----CL9C-------------------------
INT E0 - REAL/32 - "P_PDADR" - GET ADDRESS OF PROCESS DESCRIPTOR
  CL = 9Ch
Return: ES:AX -> caller's process descriptor (see #04026)
  may destroy SI,DI
Note: This function is also called by the DR DOS MEM utility when
    INT 21/AX=4451h returns either 1463h (MDOS 5.00), 1465h (MDOS 5.01),
    or 1466h (MDOS 5.10).
SeeAlso: INT E0/CL=8Fh,INT E0/CL=90h
--------O-E0----CL9D-------------------------
INT E0 - REAL/32 - "P_ABORT" - ABORT PROCESS BY NAME OR DESCRIPTOR
  CL = 9Dh
  DS:DX -> Abort Parameter Block (see #04050)
Return: AX = status
```

```
  may destroy SI,DI,DS
SeeAlso: INT E0/CL=8Fh,INT E0/CL=9Ch

Format of REAL/32 Abort Parameter Block:
Offset  Size  Description (Table 04050)
 00h  WORD  process descriptor of process to abort, or 0000h
 02h  WORD  termination code
 04h  BYTE  default console number
 05h  BYTE  reserved (0)
 06h  8 BYTEs name of process to abort, if descriptor at offset 00h is 0000h
--------O-E0----CL9E-------------------------
INT E0 - REAL/32 - "L_ATTACH" - ATTACH TO DEFAULT LIST DEVICE
  CL = 9Eh
Return: AX = status
  may destroy SI,DI
SeeAlso: INT E0/CL=92h,INT E0/CL=9Fh,INT E0/CL=A0h,INT E0/CL=A1h
--------O-E0----CL9F-------------------------
INT E0 - REAL/32 - "L_DETACH" - DETACH FROM DEFAULT LIST DEVICE
  CL = 9Fh
Return: AX = status
  may destroy SI,DI
SeeAlso: INT E0/CL=92h,INT E0/CL=9Eh
--------O-E0----CLA0-------------------------
INT E0 - REAL/32 - "L_SET" - SET DEFAULT LIST DEVICE
  CL = A0h
  DX = ID of list (printer) device
Return: AX = status (0000h success, FFFFh if invalid printer number)
  may destroy SI,DI
SeeAlso: INT E0/CL=92h,INT E0/CL=9Eh,INT E0/CL=A4h
--------O-E0----CLA1-------------------------
INT E0 - REAL/32 - "L_CATTACH" - CONDITIONALLY ATTACH TO DEFAULT LIST DEVICE
  CL = A1h
Return: AX = status (00h = successful, FFh = unable to attach)
  may destroy SI,DI
Desc: attach the default list device to the calling process only if it is
    currently available
SeeAlso: INT E0/CL=92h,INT E0/CL=9Eh,INT E0/CL=A0h
--------O-E0----CLA2-------------------------
INT E0 - REAL/32 - "C_CATTACH" - CONDITIONALLY ATTACH TO DEFAULT CONSOLE
  CL = A2h
Return: AL = status (FFh = console attached to another process)
```

```
Note: if the console is currently owned by another process, this function
    will return an error code instead of attaching or waiting
SeeAlso: INT E0/CL=93h,INT E0/CL=94h,INT E0/CL=99h,INT E0/CL=92h
--------O-E0----CLA3-------------------------
INT E0 - ConcCP/M,DR Multiuser DOS,REAL/32 - "S_OSVER" - GET OS VERSION
  CL = A3h
Return: AX = operating system version (see #04023) (see also INT 21/AX=4451h)
SeeAlso: INT E0/CL=0Ch,INT 21/AX=4451h
--------O-E0----CLA4-------------------------
INT E0 - REAL/32 - "L_GET" - GET DEFAULT LIST (PRINTER) DEVICE
  CL = A4h
Return: AL = number of default list device
SeeAlso: INT E0/CL=A0h
--------O-E0----CLA5-------------------------
INT E0 - DR Multiuser DOS - "A_ATTACH" - ATTACH AUX DEVICE
  CL = A5h
Return: nothing (DR)
  AX = status (REAL/32)
Desc: attaches the default auxiliary device to the calling process unless
    it is already attached to another process, in which case the call
    blocks until the device becomes available
Notes:  this call should be used before attempting to read or write from
    the AUX device; however, the I/O calls internally call this function
    to ensure device ownership
  also supported by REAL/32
SeeAlso: INT E0/CL=03h,INT E0/CL=04h,INT E0/CL=A6h,INT E0/CL=A7h,INT E0/CL=A8h
SeeAlso: INT E0/CL=ACh,INT E0/CL=ADh,INT E0/CL=B0h
--------O-E0----CLA6-------------------------
INT E0 - DR Multiuser DOS - "A_DETACH" - DETACH FROM AUX DEVICE
  CL = A6h
Return: AX = status
      0000h successfully detached
      FFFFh detach failed
  CX = error code
Note: also supported by REAL/32
SeeAlso: INT E0/CL=93h,INT E0/CL=A5h,INT E0/CL=A7h
--------O-E0----CLA7-------------------------
INT E0 - DR Multiuser DOS - "A_CATTACH" - CONDITIONALLY ATTACH TO AUX DEVICE
  CL = A7h
Return: AX = status
      0000h attached
```

```
      FFFFh unable to attach
Desc: attaches the default auxiliary device to the calling process if it is
    available
Notes:  does not block if the device is already in use
  also supported by REAL/32
SeeAlso: INT E0/CL=A5h,INT E0/CL=A6h,INT E0/CL=A8h,INT E0/CL=B0h
--------O-E0----CLA8-------------------------
INT E0 - DR Multiuser DOS - "A_SET" - SET DEFAULT AUX DEVICE NUMBER
  CL = A8h
  DL = auxiliary device number
Return: AX = status
      0000h successful
      FFFFh failed
  CX = error code
Desc: specify which physical device will become AUX
SeeAlso: INT E0/CL=A5h,INT E0/CL=A9h
--------O-E0----CLA9-------------------------
INT E0 - DR Multiuser DOS - "A_GET" - GET DEFAULT AUX DEVICE NUMBER
  CL = A9h
Return: AL = current default auxiliary device number
Desc: determine which physical device is currently AUX
SeeAlso: INT E0/CL=A8h
--------O-E0----CLAC-------------------------
INT E0 - DR Multiuser DOS - "A_READBLK" - READ STRING FROM AUX DEVICE
  CL = ACh
  DS:DX -> character control block (CHCB) (see #04051,#04052)
Return: AX = number of characters read
Desc: read characters from the default auxiliary (AUXn:) device into a buffer
    until the buffer is full or the device is no longer ready
Notes:  if the device is initially not ready, blocks until at least one
    character has been read
  if another process owns AUX, this call blocks until the device becomes
    available
  also supported by REAL/32
SeeAlso: INT E0/CL=03h,INT E0/CL=A5h,INT E0/CL=ADh

Format of DR Multiuser DOS character control block (CHCB):
Offset  Size  Description (Table 04051)
 00h  DWORD pointer to character buffer
 04h  WORD  length of character buffer
Note: this version of the structure is used by REAL/32 in real mode
```

```
SeeAlso: #04059

Format of REAL/32 protected-mode Character Control Block (CHCB):
Offset  Size  Description (Table 04052)
 00h  DWORD 32-bit offset of character buffer
 04h  WORD  selector for character buffer
 06h  WORD  length of character buffer
SeeAlso: #04051
--------O-E0----CLAD-------------------------
INT E0 - DR Multiuser DOS - "A_WRITEBLK" - WRITE STRING TO AUX DEVICE
  CL = ADh
  DS:DX -> character control block (see #04051,#04052)
Return: AX = number of characters written
Notes:  does not return until at least one character has been written
  also supported by REAL/32
SeeAlso: INT E0/CL=04h,INT E0/CL=A5h,INT E0/CL=ACh
--------O-E0----CLB0-------------------------
INT E0 - DR Multiuser DOS - "A_CONFIG" - GET/SET AUX DEVICE PARAMETERS
  CL = B0h
  DX:DX -> AUX device parameter block (see #04053)
Return: AX = status
      0000h successful
    parameter block updated
      FFFFh failed
    CX = error code
Note: also supported by REAL/32
SeeAlso: INT E0/CL=A5h,INT E0/CL=B1h

Format of DR Multiuser DOS AUX device parameter block:
Offset  Size  Description (Table 04053)
 00h  BYTE  function (00h get, 01h set)
 01h  BYTE  baud rate (see #04055) FFh = don't change/unknown
 02h  BYTE  parity (see #04054)
 03h  BYTE  stop bits (00h one, 01h 1.5, 02h two, FFh unknown/don't change)
 04h  BYTE  data bits (05h-08h or FFh unknown/don't change)
 05h  BYTE  handshake (00h none, 01h DTS/DSR, 02h RTS/CTS, 04h XON/XOFF,
    FFh unknown/don't change)
 06h  BYTE  XON character, FFh unknown/don't change
 07h  BYTE  XOFF character, FFh unknown/don't change

(Table 04054)
```

```
Values for DR Multiuser DOS AUX parity:
 00h  none
 01h  odd
 02h  none
 03h  even
 04h  stick parity bit
 FFh  don't change/unknown
SeeAlso: #04053,#04055

(Table 04055)
Values for DR Multiuser DOS AUX baud rate:
 00h  50 baud
 01h  62.5 baud
 02h  75 baud
 03h  110 baud
 04h  134.5 baud
 05h  150 baud
 06h  200 baud
 07h  300 baud
 08h  600 baud
 09h  1200 baud
 0Ah  1800 baud
 0Bh  2000 baud
 0Ch  2400 baud
 0Dh  3600 baud
 0Eh  4800 baud
 0Fh  7200 baud
 10h  9600 baud
 11h  19200 baud
 12h  38400 baud
 13h  56000 baud
 14h  76800 baud
 15h  115200 baud
SeeAlso: #04053,#04054
--------O-E0----CLB1-------------------------
INT E0 - DR Multiuser DOS - "A_CONTROL" - GET/SET AUX CONTROL PARAMETERS
  CL = B1h
  DS:DX -> AUX device control block (see #04056)
Return: AX = status
      0000h successful
    control block updated
```

```
      FFFFh failed
    CX = error code
Note: also supported by REAL/32
SeeAlso: INT E0/CL=B0h,INT E0/CL=B2h

Format of DR Multiuser DOS AUX device control block:
Offset  Size  Description (Table 04056)
 00h  BYTE  function (00h get, 01h set)
 01h  BYTE  DTR state (00h low, 01h high, FFh unknown/don't change)
 02h  BYTE  RTS state (00h low, 01h high, FFh unknown/don't change)
 03h  BYTE  DSR state (00h low, 01h high, FFh unknown/don't change)
 04h  BYTE  CTS state (00h low, 01h high, FFh unknown/don't change)
 05h  BYTE  DCD state (00h low, 01h high, FFh unknown/don't change)
 06h  BYTE  RI state (00h inactive, 01h active, FFh unknown/don't change)
--------O-E0----CLB2-------------------------
INT E0 - DR Multiuser DOS - "A_BREAK" - SEND BREAK TO AUX DEVICE
  CL = B2h
  DX = duration of break in system ticks (0001h-FFFFh)
Return: AX = status
      0000h successful
    break signal completed
      FFFFh failed
    CX = error code
Notes:  if the AUX device is currently owned by another process, this call will
    block until the device becomes available
  also supported by REAL/32
SeeAlso: INT E0/CL=A5h,INT E0/CL=B1h
--------O-E0----CLB3-------------------------
INT E0 R - REAL/32 - "MP_ALLOC" - ALLOCATE PHYSICAL MEMORY
  CL = B3h
  DX = number of 4K pages to allocate
Return: AX = number of first 4K page allocated, or FFFFh on error
Note: memory allocated with this function is not automatically freed when
    the process terminates
SeeAlso: INT E0/CL=B4h,INT E0/CL=B5h
--------O-E0----CLB4-------------------------
INT E0 R - REAL/32 - "MP_FREE" - DEALLOCATE PHYSICAL MEMORY
  CL = B4h
  DX = number of the physical page to free
Return: AX = status (0000h successful, FFFFh error)
SeeAlso: INT E0/CL=B3h
```

--------O-E0----CLB5-------------------------
INT E0 R - REAL/32 - "MP_MAP" - MAP PHYSICAL MEMORY
  CL = B5h
  DS:DX -> Memory Physical Parameter Block (see #04057)
Return: AX = status (0000h successful, FFFFh error)
  may destroy SI,DI,DS
SeeAlso: INT E0/CL=B3h

Format of REAL/32 Memory Physical Parameter Block:
Offset Size Description (Table 04057)
 00h WORD 4K page number in first megabyte to be mapped
 02h WORD number of page to map into above page, or 0000h to unmap
 04h WORD window number (0000h, as only one window currently supported)
 06h WORD reserved
Note: all users of the system share the window, so applications should not
    use it for arguments to system calls, attempt to read/write to/from
    files via the window, etc.
--------O-E0----CLB6-------------------------
INT E0 - REAL/32 - "Q_NCREAD" - CONDITIONALLY NONDESTRUCTIVELY READ QUEUE
  CL = B6h
  DS:DX -> Queue Parameter Block (QPB) (see #04045,#04044)
Return: AX = status
SeeAlso: INT E0/CL=89h,INT E0/CL=8Ah
--------O-E0----CLB7-------------------------
INT E0 - REAL/32 - "DEV_TWAITFLAG" - TIMED WAIT ON SYSTEM FLAG
  CL = B7h
  DS:DX -> Flag Parameter Block (FPB) (see #04058)
Return: AX = status
  may destroy SI,DI,DS
Note: if the flag was already set, this function returns immediately
SeeAlso: INT E0/CL=84h,INT E0/CL=85h

Format of REAL/32 Flag Parameter Block (FPB):
Offset Size Description (Table 04058)
 00h WORD number of system flag on which to wait
 02h WORD maximum number of system ticks (see INT E0/CL=9Ah) to wait
--------O-E0----CLB8-------------------------
INT E0 - REAL/32 - "DEV_MAP" - GET/SET SERIAL/PARALLEL PORT MAPPINGS
  CL = B8h
  DS:DX -> Device Map Parameter Block (DEVPB) (see #04059)
Return: nothing???

```
  may destroy SI,DI,DS

Format of REAL/32 Device Map Parameter Block (DEVPB):
Offset  Size  Description (Table 04059)
 00h  WORD  direction (00h = get current mapping, 01h = set new mapping)
 02h  4 BYTEs physical device IDs which map into LPT1 - LPT4
 06h  4 BYTEs physical device IDs which map into COM1 - COM4
Note: LPT4 and COM3/COM4 are reserved on early versions of Multiuser DOS,
    but are supported from at least CCI Multiuser DOS v7.22 onward
--------O-E0----CLB9-------------------------
INT E0 - REAL/32 - "L_TIMEOUT" - SET LIST DEVICE TIMEOUT
  CL = B9h
  DX = timeout value in system ticks (see #04048)
    0000h-FFEFh = new number of system ticks
    FFFDh = get current value without changing
    FFFEh = start timeout count
    FFFFh = never timeout
Return: AX = status, or current timeout value if DX=FFFDh on entry
  may destroy SI,DI
SeeAlso: INT E0/CL=BAh
--------O-E0----CLBA-------------------------
INT E0 - REAL/32 - "A_TIMEOUT" - SET AUX DEVICE TIMEOUT
  CL = BAh
  DX = timeout value in system ticks (see #04048)
    0000h-FFEFh = new number of system ticks
    FFFDh = get current value without changing
    FFFEh = start timeout count
    FFFFh = never timeout
Return: AX = status, or current timeout value if DX=FFFDh on entry
  may destroy SI,DI
SeeAlso: INT E0/CL=B9h
--------O-E0----CLBD-------------------------
INT E0 - DR Multiuser DOS - "P_DELAY" - DELAY EXECUTION
  CL = BDh
  DX = delay in system ticks (typically 16.6 ms/tick)
Return: after the delay elapses
  no results
Notes:  the length of a system tick is installation-dependent (typically
    1/50 or 1/60 second); the length may be determined by reading the
    TICKSPERSEC value from the system data segment
  the actual delay before the process is rescheduled to run may be up to
```

```
    one tick longer than requested; the delay between rescheduling and
    actual execution cannot be predicted if higher-priority processes
    are awaiting a turn at the CPU
SeeAlso: INT 15/AH=86h,INT 1A/AX=FF01h,INT 2F/AX=1224h,INT 62/AX=0096h
--------g-E00000-----------------------------
INT E0 - PCROBOTS v1.41 - "SWAPTASK" - END CURRENT ROBOT'S TURN
  AX = 0000h
Return: nothing
Program: PCROBOTS is P.D. Smith's adaptation of Tom Poindexter's CROBOTS, in
    which specially-written .COM or .EXE programs form robots battling
    each other in a user-defined arena
--------g-E00001-----------------------------
INT E0 - PCROBOTS v1.41 - "MOVEMENT" - START MOVING
  AX = 0001h
  BX = speed (0-maximum for robot)
  CX = direction (0-359 degrees)
Return: nothing
Notes:  the speed will change to the specified value at the maximum
    acceleration the robot is capable of; if the robot is already moving
    faster than its maximum maneuverability speed, it will not be able
    to change direction
  this call also terminates the current robot's turn
SeeAlso: AX=0000h,AX=0002h,AX=0003h
--------g-E00002-----------------------------
INT E0 - PCROBOTS v1.41 - "SCAN" - SCAN FOR OTHER ROBOTS IN THE GIVEN DIRECTION
  AX = 0002h
  BX = direction (0-359 degrees)
  CX = resolution (0-45 degrees)
Return: AX = status
      FFFFh if nothing detected
      else robot ID (0-19)
    BX = range to detected robot
Notes:  the scan searches within CX degrees to either side of the specified
    direction
  the scanner will see right through walls, but shells will not pass
    through walls
  this call also terminates the current robot's turn
SeeAlso: AX=0000h,AX=0001h,AX=0003h
--------g-E00003-----------------------------
INT E0 - PCROBOTS v1.41 - "SHOOT" - FIRE A SHELL AT ANOTHER ROBOT
  AX = 0003h
```

```
  BX = direction (0-359 degrees)
  CX = range (0-700)
Return: AX = status (0000h not fired, else ID of shell fired)
Notes:  up to seven shells may be in flight for a robot at one time; the cannon
    takes 50 ticks to reload
  this call also terminates the current robot's turn
SeeAlso: AX=0000h,AX=0001h,AX=0002h,AX=002Ch
--------g-E00010-----------------------------
INT E0 - PCROBOTS v1.41 - "GETXY" - GET ROBOT'S CURRENT POSITION
  AX = 0010h
Return: BX = current X coordinate (0-999)
  CX = current Y coordinate (0-999)
--------g-E00011-----------------------------
INT E0 - PCROBOTS v1.41 - "TRANSMIT" - SEND DATA TO ANOTHER ROBOT
  AX = 0011h
  BX = target robot ID
  CX = data to be sent
Return: AX = status (0000h data could not be sent, 0001h data sent)
Note: this call costs one unit of battery power
--------g-E00012-----------------------------
INT E0 - PCROBOTS v1.41 - "RECEIVE" - GET DATA FROM OTHER ROBOTS
  AX = 0012h
Return: AX = status
      0000h no data available
      0001h data retrieved
    BX = sender's ID
    CX = data
Note: each robot has a 20-word receive FIFO; if the FIFO is full, other
    robots will be unable to send more data until some is read
--------g-E00013-----------------------------
INT E0 - PCROBOTS v1.41 - "DAMAGE" - DETERMINE HOW MUCH DAMAGE SUSTAINED
  AX = 0013h
Return: BX = damage status
Note: the initial value depends on configuration, but is typically 100; as
    the robot is damaged, it decreases
--------g-E00014-----------------------------
INT E0 - PCROBOTS v1.41 - "SPEED" - DETERMINE HOW FAST ROBOT IS MOVING
  AX = 0014h
Return: BX = current speed
--------g-E00015-----------------------------
INT E0 - PCROBOTS v1.41 - "BATTERY" - DETERMINE HOW MUCH BATTERY POWER LEFT
```

```
  AX = 0015h
Return: BX = current battery charge
Note: the battery starts off with 1000 units of charge, and is constantly
    being charged by solar panels and constantly discharged by motion;
    the battery is charged at 4 units per turn and discharged at
    0.1*speed units per turn.
--------g-E00016-----------------------------
INT E0 - PCROBOTS v1.41 - "TICKS" - DETERMINE HOW LONG SINCE GAME STARTED
  AX = 0016h
Return: BX:CX = number of game ticks elapsed (not related to real time)
--------g-E00017-----------------------------
INT E0 - PCROBOTS v1.41 - "L_SIN" - GET SCALED SINE OF AN ANGLE
  AX = 0017h
  BX = angle (0-359 degrees)
Return: BX:CX = 100000*sine of angle
SeeAlso: AX=0018h,AX=0019h,AX=001Ah,AX=001Bh
--------g-E00018-----------------------------
INT E0 - PCROBOTS v1.41 - "L_COS" - GET SCALED COSINE OF AN ANGLE
  AX = 0018h
  BX = angle (0-359 degrees)
Return: BX:CX = 100000*cosine of angle
SeeAlso: AX=0017h,AX=0019h,AX=001Ah
--------g-E00019-----------------------------
INT E0 - PCROBOTS v1.41 - "L_TAN" - GET SCALED TANGENT OF AN ANGLE
  AX = 0019h
  BX = angle (0-359 degrees)
Return: BX:CX = 100000*tangent of angle
SeeAlso: AX=0017h,AX=0018h,AX=001Ah
--------g-E0001A-----------------------------
INT E0 - PCROBOTS v1.41 - "L_ATAN" - GET ANGLE GIVEN SCALED TANGENT
  AX = 001Ah
  BX:CX = 100000*tangent of an angle
Return: AX = angle (-90 to +90 degrees)
SeeAlso: AX=0017h,AX=0018h,AX=0019h
--------g-E0001B-----------------------------
INT E0 - PCROBOTS v1.41 - "SQRT" - DETERMINE SQUARE ROOT OF A NUMBER
  AX = 001Bh
  BX:CX = value
Return: BX:CX = square root
SeeAlso: AX=0017h
--------g-E0001C-----------------------------
```

```
INT E0 - PCROBOTS v1.41 - "SET_PATTERN" - SPECIFY ROBOT'S DISPLAY IMAGE
  AX = 001Ch
  BX:CX -> pattern array
Return: nothing
Note: the pattern array consists of five bytes, the low five bits of each
    specifying the bit pattern for one line of the robot's screen display
--------g-E0001D-----------------------------
INT E0 - PCROBOTS v1.41 - "DEBUG_FLAG" - SET/CLEAR MARKERS NEXT TO ROBOT'S NAME
  AX = 001Dh
  BX = flag number (0 or 1)
  CX = new value (0 reset, 1 set)
Return: nothing
Program: PCROBOTS is P.D. Smith's adaptation of Tom Poindexter's CROBOTS, in
    which specially-written .COM or .EXE programs form robots battling
    each other in a user-defined arena
Note: the two flag markers may be used for any purpose, typically for
    debugging to provide a visual display of progress
--------g-E0001E-----------------------------
INT E0 - PCROBOTS v1.41 - "BUY_ARMOUR" - BUY OR SELL ARMOR FOR ROBOT
  AX = 001Eh
  BX = number of armor units to buy (negative to sell)
Return: nothing
Note: each armor unit is worth 50 battery units; the robot's armor rating
    will not go above its initial rating, so attempts to purchase more
    will waste battery units
SeeAlso: AX=001Fh
--------g-E0001F-----------------------------
INT E0 - PCROBOTS v1.41 - "BUY_SHELLS" - BUY ADDITIONAL CANNON SHELLS
  AX = 001Fh
  BX = number of shells to buy
Return: nothing
Note: each shell costs ten battery units
SeeAlso: AX=001Eh,AX=0020h
--------g-E00020-----------------------------
INT E0 - PCROBOTS v1.41 - "SHELLS LEFT" - DETERMINE HOW MANY SHELLS ROBOT HAS
  AX = 0020h
Return: BX = number of shells remaining
SeeAlso: AX=001Fh
--------g-E00021-----------------------------
INT E0 - PCROBOTS v1.41 - "GET LOCAL MAP"
  AX = 0021h
```

```
  BX:CX -> 81-byte buffer for map (see #04060)
Return: buffer filled with 9x9 area of map centered on robot's position

(Table 04060)
Values for PCROBOTS map squares:
 2Eh '.' empty square
 44h 'D' damaging trap
 52h 'R' refueling point
 58h 'X' wall
--------g-E00022----------------------------
INT E0 - PCROBOTS v1.41 - "INVISIBILITY" - CONTROL ROBOT'S INVISIBILITY DEVICE
  AX = 0022h
  BX = new state (0000h become visible, 0001h become invisible)
Return: nothing
Notes:  this function has no effect if the robot is not capable of invisibility
  the robot can only stay invisible for 100 turns, after which it will
    automatically become visible; it must also be remain visible for
    as many turns as it was invisible before it can turn invisible
    again
SeeAlso: AX=0024h,AX=0080h
--------g-E00023----------------------------
INT E0 - PCROBOTS v1.41 - "GET_SHELL_STATUS" - FIND OUT WHAT HAPPENED TO SHELL
  AX = 0023h
Return: BX = status of last shell to land
      0000h missed completely
      0001h hit a wall
      0002h hit a robot within 50-square radius
      0003h hit a robot within 25-square radius
      0004h hit a robot within 5-square radius
--------g-E00024----------------------------
INT E0 - PCROBOTS v1.41 - "IS_INVISIBLE" - DETERMINE WHETHER ROBOT IS INVISIBLE
  AX = 0024h
Return: BX = visibility (0000h visible, 0001h invisible)
SeeAlso: AX=0022h,AX=0080h
--------g-E00025----------------------------
INT E0 - PCROBOTS v1.41 - "L_ATAN2" - GET ARCTANGENT
  AX = 0025h
  BX = Y
  CX = X
Return: AX = angle (arctangent of Y/X)
--------g-E00026----------------------------
```

```
INT E0 - PCROBOTS v1.41 - "GET_ROBOT_ID" - DETERMINE CURRENT ROBOT'S IDENTIFIER
  AX = 0026h
Return: AX = robot ID
--------g-E00027----------------------------
INT E0 - PCROBOTS v1.41 - "REGISTER_IFF" - REGISTER FRIEND/FOE IDENT STRING
  AX = 0027h
  BX:CX = ASCIZ IFF string
Return: nothing
Note: the IFF string may only be set once
SeeAlso: AX=0028h,AX=0029h
--------g-E00028----------------------------
INT E0 - PCROBOTS v1.41 - "CHECK_IFF" - QUERY FRIEND/FOE IDENTIFICATION STRING
  AX = 0028h
  BX = robot ID to test
Return: AX = status
      0000h IFF strings match
      0001h IFF strings differ or invalid robot ID
SeeAlso: AX=0027h
--------g-E00029----------------------------
INT E0 - PCROBOTS v1.41 - "REGISTER_NAME" - SPECIFY ROBOT'S NAME
  AX = 0029h
  BX:CX -> ASCIZ name string
Return: nothing
Note: the name may only be set once
SeeAlso: AX=0027h,AX=002Ah
--------g-E0002A----------------------------
INT E0 - PCROBOTS v1.41 - "FIND_NAME" - SEARCH FOR ROBOT WITH GIVEN NAME
  AX = 002Ah
  BX:CX -> ASCIZ name string
  DX = first ID to check
Return: AX = robot ID or FFFFh if no robot with specified name
SeeAlso: AX=0028h,AX=0029h,AX=002Bh
--------g-E0002B----------------------------
INT E0 - PCROBOTS v1.41 - "GET_TEAM_ID" - DETERMINE TEAM MEMBERSHIP OF ROBOT
  AX = 002Bh
Return: AX = team ID (0-2) or FFFFh if 'loner'
SeeAlso: AX=0029h
--------g-E0002C----------------------------
INT E0 - PCROBOTS v1.41 - "GET_ASHELL_STATUS" - FIND OUT WHAT HAPPENED TO SHELL
  AX = 002Ch
  BX = shell ID
```

```
Return: AX = status
      0000h missed completely
      0001h hit a wall
      0002h hit a robot within a 50-square radius
      0003h hit a robot within a 25-square radius
      0004h hit a robot within a 5-square radius
      0005h shell not known (too old or not yet fired)
      0006h shell still in flight
SeeAlso: AX=0003h
--------g-E0002D-----------------------------
INT E0 - PCROBOTS v1.41 - "REGISTER_X" - SELECT AUTOMATIC X POSITION UPDATES
  AX = 002Dh
  BX:CX -> X word variable
Return: AX = status (0001h OK, 0000h problem with address)
Note: after this call, PCROBOTS will automatically update the specified
    word whenever the robot moves
SeeAlso: AX=002Eh
--------g-E0002E-----------------------------
INT E0 - PCROBOTS v1.41 - "REGISTER_Y" - SELECT AUTOMATIC Y POSITION UPDATES
  AX = 002Eh
  BX:CX -> Y word variable
Return: AX = status (0001h OK, 0000h problem with address)
Note: after this call, PCROBOTS will automatically update the specified
    word whenever the robot moves
SeeAlso: AX=002Dh
--------g-E00080-----------------------------
INT E0 - PCROBOTS v1.41 - "CONFIGURE" - CUSTOMIZE ROBOT
  AX = 0080h
  BX = basic configuration (see #04061)
  CX = advanced configuration (see #04062)
Return: AX = status (0001h OK, 0000h not first call in program)
Program: PCROBOTS is P.D. Smith's adaptation of Tom Poindexter's CROBOTS, in
    which specially-written .COM or .EXE programs form robots battling
    each other in a user-defined arena
Notes:  a maximum of ten points may be allocated to the robot; if you attempt
    to allocate more, some items will be given a value of zero.  If this
    function is not called, each attribute is set to the default value
    of 2.
  if the invisibility option is chosen, the robot will start with only
    900 cannon shells instead of the default 1000
```

```
Bitfields for PCROBOTS basic configuration:
Bit(s)  Description (Table 04061)
 0-3  maximum speed (0-4 = 50,75,100,150,200)
 4-7  maneuverability (0-4 = 20%,35%,50%,75%,100%)
 8-11 cannon range (0-4 = 300,500,700,1000,1500)
 12-15  robot armor (0-4 = 50,75,100,150,200)

Bitfields for PCROBOTS advanced configuration:
Bit(s)  Description (Table 04062)
 0-2  robot acceleration (0-4 = 5,7,10,15,20)
 3  capable of invisibility
--------r-E1---------------------------------
INT E1 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
--------N-E1---------------------------------
INT E1 - PC Cluster Disk Server Information (NOT A VECTOR!)
Desc: points at a data table
SeeAlso: INT E2
--------O-E1---------------------------------
INT E1 - MP/M-86, - ALTERNATE CP/M-86 FUNCTION CALLS
  CL = function number (see #04019,#04020)
  DS,DX = parameters
Return: as appropriate for function
  CX is often the error code (see #04021)
Desc: used by some applications which alter CP/M functions while running a
    child program, to store the original INT E0 vector before
    intercepting INT E0
Notes:  This is the debugger entry to the operating system.
  This function is also supported by IMS Multiuser DOS and IMS REAL/32.
SeeAlso: #04019 at INT E0"CP/M"
----------E1---------------------------------
INT E1 - DeskMate (Tandy) - TASK DATA SEGMENTS (NOT A VECTOR!)
Desc: used to store data; the low word of the vector is the data segment for
    the first task; the high word is the data segment of the second task
    (DeskMate supports 2-way task switching between small- or
    medium-model applications)
Program: DeskMate is a proprietary GUI from Tandy distributed with several
    models of the Tandy 1000's, 2500's, 3000's, and laptops.  Retail
    and runtime versions also exist.  Some Tandy's are designed to
    boot directly into DeskMate.
```

```
SeeAlso: INT E0"DeskMate"
--------r-E2---------------------------------
INT E2 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
--------N-E2---------------------------------
INT E2 - PC Cluster Program - RELOCATED INT 1C
SeeAlso: INT 1C
--------r-E3---------------------------------
INT E3 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
--------r-E40005-----------------------------
INT E4 - Logitech Modula v2.0 - MonitorEntry
  AX = 0005h
  BX = priority
Return: nothing
SeeAlso: AX=0006h
--------r-E40006-----------------------------
INT E4 - Logitech Modula v2.0 - MonitorExit
  AX = 0006h
Return: nothing
SeeAlso: AX=0005h
--------r-E4---------------------------------
INT E4 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
----------E4---------------------------------
INT E4 - DIGITAL RESEARCH - FLAG INTERRUPT
Note: This interrupt corresponds with Concurrent CP/M-86 and is to get
    an unused flag.
SeeAlso: INT E5"DIGITAL RESEARCH"
--------r-E5---------------------------------
INT E5 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
----------E5---------------------------------
INT E5 - DIGITAL RESEARCH - FIDDS INTERRUPT
Note: This interrupt corresponds with Concurrent CP/M-86 and is for
    "attachamatic" drives
SeeAlso: INT E4"DIGITAL RESEARCH",INT E6"CP/M"
```

```
--------r-E6---------------------------------
INT E6 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
--------O-E6---------------------------------
INT E6 C - CP/M-86 v1.1 - XIOS INTERRUPT / UNKNOWN DISK DRIVE
  AX = function which accessed drive
       0000h SELDSK
       0001h READ
       0002h WRITE
       0003h HOME
  DX:BX -> parameter block (see #04063)
Return: AX = return value
Desc: called by CP/M-86 kernel when an unknown disk drive is used, which
    permits an application to provide access services
Note: This was labelled "XIOS interrupt" in later Digital Research
    documentation (CCP/M-86) and "for the version 1.0 back door".

Format of CP/M-86 unknown-drive paramter block:
Offset  Size  Description (Table 04063)
 00h  BYTE  drive number (00h = first unknown drive, etc.)
 01h  BYTE  deblocking flag (00h = normal write, 01h = directory write)
 02h  WORD  track number
 04h  WORD  sector number
 06h  DWORD data transfer address
 0Ah  BYTE  verify flag (nonzero to verify writes)
Note: in CP/M-86 v1.1, this is actually a portion of a BIOS disk data
    table which starts one byte before the address given to the INT E6
    handler; that extra byte is the current logical drive number
--------O-E600-------------------------------
INT E6 - Linux DOSEMU - INSTALLATION CHECK
  AH = 00h
Return: AX = AA55h if installed
      BH = major version number
      BL = minor version number
      CX = patchlevel
Notes:  check for the BIOS date string "02/25/93" at F000:FFF5 before
    calling this function.  In addition, the segment address of this
    vector should be F000h (for existing versions of DOSemu, the
    vector is F000h:0E60h)
SeeAlso: AH=FFh
```

```
--------O-E601--------------------------------
INT E6 - Linux DOSEMU - REGISTER DUMP
  AH = 01h
Return: nothing
SeeAlso: AH=00h
--------O-E602--------------------------------
INT E6 - Linux DOSEMU - SET I/O PORT PERMISSIONS
  AH = 02h
  BX = base I/O port address
  CX = number of consecutive I/O ports
  CF set to allow DOS to use ports
  CF clear if DOS should not be allowed to use ports
Return: nothing
SeeAlso: AH=00h
--------O-E605--------------------------------
INT E6 - Linux DOSEMU - STARTUP BANNER
  AH = 05h
Return: nothing
SeeAlso: AH=00h
--------O-E612--------------------------------
INT E6 - Linux DOSEMU - SET "HOGTHRESHOLD"
  AH = 12h
  BX = new "hogthreshold" (00h-99h)
Return: nothing
Desc: specify how much CPU time DOSEMU may use
SeeAlso: AH=00h
--------O-E622--------------------------------
INT E6 - Linux DOSEMU - GET EMS STATUS
  AH = 22h
Return: ???
SeeAlso: AH=00h
--------O-E630--------------------------------
INT E6 - Linux DOSEMU - SET BOOTDISK FLAG
  AH = 30h
  BX = new flag state (0 = false, 1 = true)
Return: nothing
SeeAlso: AH=00h
--------O-E650--------------------------------
INT E6 - Linux DOSEMU - EXECUTE UNIX COMMAND
  AH = 50h
  ES:DX -> ASCIZ Unix command
```

```
SeeAlso: AH=00h,AH=51h
--------O-E651---------------------------------
INT E6 - Linux DOSEMU - EXECUTE DOS COMMAND FROM UNIX
  AH = 51h
  ES:DX -> ASCIZ DOS command
SeeAlso: AH=00h,AH=50h
--------O-E680---------------------------------
INT E6 - Linux DOSEMU - GET CURRENT UNIX DIRECTORY
  AH = 80h
Return: ES:DX -> current Unix directory
  AX = length of current directory name
SeeAlso: AH=00h,AH=81h
--------O-E681---------------------------------
INT E6 - Linux DOSEMU - CHANGE CURRENT UNIX DIRECTORY
  AH = 81h
  ES:DX -> ASCIZ directory name
Return: nothing
SeeAlso: AH=00h,AH=80h
--------O-E6FF---------------------------------
INT E6 - Linux DOSEMU - TERMINATE
  AH = FFh
SeeAlso: AH=00h
--------r-E7---------------------------------
INT E7 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
--------r-E8---------------------------------
INT E8 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
--------r-E9---------------------------------
INT E9 - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
--------r-EA---------------------------------
INT EA - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
--------r-EB---------------------------------
INT EB - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
```

```
  BASIC.COM/BASICA.COM do not restore vector on termination
--------r-EC--------------------------------
INT EC - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC
  BASIC.COM/BASICA.COM do not restore vector on termination
--------N-EC--------------------------------
INT EC - used by Alloy NTNX
--------r-EC--------------------------------
INT EC - Exact - RUNTIME INTERFACE MULTIPLEXOR
  AX = function number (0000h to 0140h)
  STACK:  DWORD address to return to
    any arguments required by function
Return: STACK:  return address popped, but otherwise unchanged
Desc: this is the interface from applications to the runtime system by Exact
    Automatisering B.V. of the Netherlands.  By using this interrupt,
    it can provide DLL-style capabilities under MS-DOS.
Note: the interrupt handler removes the return address and flags placed on
    the stack by the INT EC, then jumps to the appropriate function
--------r-ED--------------------------------
INT ED - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
  INT 80 through INT ED are modified but not restored by Direct Access
    v4.0, and may be left dangling by other programs written with the
    same version of compiled BASIC
SeeAlso: INT EC"BASIC",INT EE"BASIC"
--------r-EE--------------------------------
INT EE - IBM ROM BASIC - used while in interpreter
Notes:  called by ROM BASIC, but pointed at IRET by BASIC.COM/BASICA.COM
  BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT ED"BASIC",INT EE"BASIC"
--------r-EF--------------------------------
INT EF - BASIC - ORIGINAL INT 09 VECTOR
Note: BASIC.COM/BASICA.COM do not restore vector on termination
SeeAlso: INT EE"BASIC",INT F0"BASIC"
--------O-EF----CX00C8----------------------
INT EF - GEM - AES INTERFACE
  CX = 00C8h
  DX = 0000h
  ES:BX -> AES parameter block (see #04064)
SeeAlso: INT EF/CX=00C9h,INT EF/CX=0473h
```

```
Format of AES parameter block:
Offset  Size  Description (Table 04064)
 00h  DWORD -> control array (see #04065)
 04h  DWORD -> global variables (15 WORDs)
 08h  DWORD -> integer input parameters
 0Ch  DWORS -> buffer for integer results
 10h  DWORD -> address (DWORD) input parameters
 14h  DWORD -> buffer for address (DWORD) results
SeeAlso: #04066

Format of AES control array:
Offset  Size  Description (Table 04065)
 00h  WORD  function number (000Ah-0084h)
 02h  WORD  number of words of integer input parameters
 04h  WORD  number of words available for integer results
 06h  WORD  number of words of address input parameters
 08h  WORD  number of words available for address results
SeeAlso: #04064
--------O-EF----CX00C9----------------------
INT EF - GEM - AES INTERFACE
  CX = 00C9h
  DX = 0000h
  ES:BX -> AES parameter block (see #04064)
SeeAlso: INT EF/CX=00C8h,INT EF/CX=0473h
--------O-EF----CX0473----------------------
INT EF - GEM - VDI INTERFACE
  CX = 0473h
  DS:DX -> GEM VDI parameter block (see #04066)
Note: if installed, one of the following ASCII signatures will appear two
    bytes after the INT EF entry point:
      "GEMVDI"  GEM VDI (but not AES) is present
      "GEMAES"  GEM/ViewMAX VDI and AES are both present
      "ViewMAX" ViewMAX VDI (only) is present
    each of the above is followed by an ASCIZ number indicating the
    version ("10" for GEM/1 AES and VDI; "20" for GEM/2,GEM/3,ViewMAX
    AES and GEM/2 VDI; "23" for GEM/3 VDI; and "1" for ViewMAX VDIs)
SeeAlso: INT EF/CX=00C8h,INT EF/CX=00C9h
Index:  installation check;GEM

Format of VDI parameter block:
```

```
Offset  Size  Description (Table 04066)
 00h  DWORD -> control array (see #04067)
 08h  DWORD -> integer parameters
 0Ch  DWORD -> (x,y) pair parameters
 10h  DWORD -> integer results
 14h  DWORD -> (x,y) pair results
SeeAlso: #04064

Format of VDI control array:
Offset  Size  Description (Table 04067)
 00h  WORD  (call) function number (0001h-0084h)
 02h  WORD  (call) number of words of pair parameters
 04h  WORD  (ret) number of words of pair results
 06h  WORD  (call) number of words of integer parameters
 08h  WORD  (ret) number of words of integer results
 0Ah  WORD  subfunction number
 0Ch  WORD  graphics handle
 0Eh  DWORD (call) -> pointer parameter
 12h  DWORD (ret) -> pointer result
SeeAlso: #04066
--------r-F0---------------------------------
INT F0 - BASICA.COM, GWBASIC, compiled BASIC - ORIGINAL INT 08 VECTOR
Note: BASICA.COM does not restore vector on termination
SeeAlso: INT EF"BASIC"
--------*-F1---------------------------------
INT F1 - reserved for user interrupt
--------s-F1---------------------------------
INT F1 - SPEECH.COM - CONVERT TEXT STRING TO SPEECH
  DS:BX -> '$'-terminated text string
Return: nothing
Program: SPEECH.COM is a resident text-to-speech converter by Douglas Sisco
--------s-F1---------------------------------
INT F1 - Andy C. McGuire SPEECH.COM/SAY.COM
SeeAlso: INT F2"SPEECH"
--------U-F1---------------------------------
INT F1 - AQUEDUCT, PIPELINE - GET DATA AREA ADDRESS
Return: AX:BX -> data area
Program: AQUEDUCT and PIPELINE are TSRs by James W. Birdsall to connect COM1
    and COM2 in software
Note: The installation check consists of testing for the following signature
    immediately preceding the interrupt handler: "JWBtvv" where 't' is
```

```
    either "A" for AQUEDUCT or "P" for PIPELINE and "vv" is a two-digit
    version number
Index:  installation check;AQUEDUCT|installation check;PIPELINE
--------N-F1---------------------------------
INT F1 - NetWare Remote Boot - INSTALLATION CHECK (NOT A VECTOR!)
Note: if this vector contains the value 5774654Eh ("NetW"), the remote boot
    BIOS extension is active, and access to the floppy disk is redirected
    to an image file in the server's SYS:LOGIN directory
SeeAlso: INT F2"NetWare",INT F3"NetWare",INT F4"NetWare"
--------v-F1---------------------------------
INT F1 - VIRUS - "Violetta" - ???
Note: used but not chained by virus
SeeAlso: INT E0"VIRUS",INT FF"VIRUS"
----------F101-------------------------------
INT F1 - Common ISDN API (CAPI) v1.1 - "API-REGISTER" - INITIALIZE CAPI
  AH = 01h
  ES:BX -> buffer for CAPI's use (refer to note below)
  CX = minimum number of pending messages
  DX = maximum simultaneous Level 3 connections
  SI = maximum concurrent received B3 data blocks
  DI = maximum B3 data block size
Return: AX = CAPI-assigned application ID
      0000h on error
    BX = error number
        1001h registration error
Range:  INT 00 to INT FF, selectable by program parameter
Notes:  the caller is required to provide at least 512 bytes of stack space
  the CAPI interrupt handler begins with a header which is nearly
    identical to the IBM Interrupt Sharing Protocol header
    (see #02568 at INT 2D"AMIS"), except that the short jump instruction
    to a hardware reset handler at offset 09h is replaced by the
    signature bytes "IA"
  the maximum length of a message is fixed at 180 bytes; the standard
    document suggests using CX=10, DI=1, SI=7, and DI=130 for
    applications which use only a single connection and standard
    protocols
  the total size of the application-provided buffer must be at least
    180*CX + DX*SI*DI bytes
SeeAlso: AH=02h,INT F1/AL=01h
Index:  installation check;Common ISDN API
----------F1--01-----------------------------
```

```
INT F1 - Common ISDN API (CAPI) v2.0 - "CAPI_REGISTER" - INITIALIZE CAPI
  AL = 01h
  AH = CAPI version number * 10 (14h for v2.0)
  ES:BX -> buffer for CAPI's use (refer to note below)
  CX = number of bytes for message buffer
  DX = maximum simultaneous logical (Level 3) connections
  SI = maximum concurrent received B3 data blocks (min. 2)
  DI = maximum B3 data block size (up to 2048 bytes)
Return: AX = CAPI-assigned application ID
      0000h on error
    BX = error number
        1001h registration error
Range:  INT 00 to INT FF, selectable by program parameter
Notes:  the caller is required to provide at least 512 bytes of stack space
  the CAPI interrupt handler begins with a header (see #04068) which is
    nearly identical to the IBM Interrupt Sharing Protocol header
    (see #02568 at INT 2D"AMIS"), except that the short jump instruction
    to a hardware reset handler at offset 09h is zeroed out and the
    entire header is inexplicably shortened by one byte
  the standard document suggests using 1024 + (1024*DX) bytes for the
    message buffer for typical applications
  the total size of the application-provided buffer must be at least
    CX + DX*SI*DI bytes
SeeAlso: AH=01h,INT F1/AL=02h
Index:  installation check;Common ISDN API

Format of CAPI v2.0 interrupt handler entry point:
Offset  Size  Description (Table 04068)
 00h  2 BYTEs short jump to actual start of interrupt handler, immediately
      following this data block (EBh 0Fh)
 02h  DWORD address of next handler in chain
 06h  WORD  signature 424Bh
 08h  BYTE  EOI flag (80h)
    80h primary hardware interrupt handler (will issue EOI)
 09h  2 BYTEs reserved (0)
    (is short jump to hardware reset routine in ISP header)
 0Bh  4 BYTEs signature "CAPI"
 0Fh  2 BYTEs two-digit CAPI version number in ASCII ('20')
SeeAlso: #02568 at INT 2D
----------F102------------------------------
INT F1 - Common ISDN API (CAPI) v1.1 - "API-RELEASE" - UNREGISTER FROM CAPI
```

```
  AH = 02h
  DX = application ID (see AH=01h)
Return: AX = status (0000h,1002h) (see #04069)
Notes:  the caller is required to provide at least 512 bytes of stack space
SeeAlso: AH=01h,INT F1/AL=02h

(Table 04069)
Values for CAPI v1.1 error code:
 0000h  successful
 1001h  registration error
 1002h  invalid application ID
 1003h  message too small or incorrectly coded message number
 1004h  invalid command or subcommand
 1005h  message queue full
 1006h  message queue empty
 1007h  message(s) lost due to queue overflow
 1008h  error uninstalling
SeeAlso: #04070
----------F1--02------------------------------
INT F1 - Common ISDN API (CAPI) v2.0 - "CAPI_RELEASE" - UNREGISTER FROM CAPI
  AL = 02h
  AH = CAPI version number * 10 (14h for v2.0)
  DX = application ID (see INT F1/AL=01h)
Return: AX = status (0000h,11xxh) (see #04070)
Notes:  the caller is required to provide at least 512 bytes of stack space
SeeAlso: AH=02h,INT F1/AL=01h,INT F1/AL=03h

(Table 04070)
Values for CAPI v2.0 error code:
 0000h  successful
 1001h  too many applications
 1002h  logical block size too small (must be at least 128 bytes)
 1003h  buffer > 64K
 1004h  message buffer too small (minimum 1024 bytes)
 1005h  too many logical connections requested
 1006h  reserved
 1007h  could not register because CAPI busy, try again
 1008h  OS resource unavailable (out of memory, etc.)
 1009h  COMMON-ISDN-API not installed
 100Ah  controller does not support external equipment
 100Bh  controller supports only external equipment
```

```
 1101h  invalid application ID
 1102h  illegal command or subcommand, or message too short
 1103h  message queue full
 1104h  queue empty
 1105h  queue overflowed (message lost)
 1106h  unknown notification parameter
 1107h  could not accept message because CAPI busy, try again
 1108h  OS resource unavailable (out of memory, etc.)
 1109h  COMMON-ISDN-API not installed
 110Ah  controller does not support external equipment
 110Bh  controller supports only external equipment
 2001h  message not supported in current state
 2002h  illegal controller/PLCI/NCCI
 2003h  out of PLCI
 2004h  out of NCCI
 2005h  out of LISTEN
 2006h  out of FAX resources (T.30 protocol)
 2007h  illegal message parameter coding
 3001h  unsupported B1 protocol
 3002h  unsupported B2 protocol
 3003h  unsupported B3 protocol
 3004h  unsupported B1 protocol parameter
 3005h  unsupported B2 protocol parameter
 3006h  unsupported B3 protocol parameter
 3007h  unsupported B protocol combination
 3008h  NCPI not supported
 3009h  unknown CIP value
 300Ah  unsupported flags (reserved bits set)
 300Bh  unsupported facility
 300Ch  data length not supported by current protocol
 300Dh  reset procedure not supported by current protocol
SeeAlso: #04069
----------F103------------------------------
INT F1 - Common ISDN API (CAPI) v1.1 - "API-PUT-MESSAGE"
  AH = 03h
  DX = application ID (see AH=01h)
  ES:BX -> message to be sent (see #04071)
Return: AX = status (0000h,1002h,1003h,1004h,1005h) (see #04069)
Notes:  the caller is required to provide at least 512 bytes of stack space
  the message buffer may be reused as soon as this call returns
SeeAlso: AH=01h,AH=04h,INT F1/AL=03h
```

```
Format of CAPI message:
Offset  Size  Description (Table 04071)
 00h  WORD  total message length, including header
 02h  WORD  application ID (see AH=01h)
 04h  BYTE  command (see #04072,#04073)
 05h  BYTE  subcommand (see #04072,#04073)
 06h  WORD  message sequence number
    0000h-7FFFh messages from application to CAPI (and replies)
    8000h-FFFFh messages from CAPI to application (and replies)
 08h  var message data (max 172 bytes for v1.1 only)

(Table 04072)
Values for CAPI v1.1 message command/subcommand:
Cmd/SubCmd  Name      Description
 01h/00h    RESET-B3-REQ  request resetting of a Level 3 connection
 01h/01h    RESET-B3-CONF confirm Level 3 connection reset
 01h/02h    RESET-B3-IND  indication from CAPI that Level 3 conn. reset
 01h/03h    RESET-B3-RESP confirm receipt of RESET-B3-IND
 02h/00h    CONNECT-REQ   establish B-channel connection
 02h/01h    CONNECT-CONF  confirm start of connection establishment
 02h/02h    CONNECT-IND   indication from CAPI of incoming connection
 02h/03h    CONNECT-RESP  accept incoming connection
 03h/02h    CONNECT-ACTIVE-IND  indication that B-channel connection complete
 03h/03h    CONNECT-ACTIVE-RESP confirm connection-complete indication
 04h/00h    DISCONNECT-REQ  request shutdown of B-channel connection
 04h/01h    DISCONNECT-CONF confirm shutdown of B-channel connection
 04h/02h    DISCONNECT-IND  indication that B-channel is shutting down
 04h/03h    DISCONNECT-RESP confirm that application knows of shutdown
 05h/00h    LISTEN-REQ    enable indication of incoming connections
 05h/01h    LISTEN-CONF   confirm enabling of incoming conn. indication
 06h/00h    GET-PARAMS-REQ  request B-channel parameters
 06h/01h    GET-PARAMS-CONF return B-channel parameters
 07h/00h    INFO-REQ    set B-channel info to be signalled to app
 07h/01h    INFO-CONF   confirm B-channel info signalling
 07h/02h    INFO-IND    signal B-channel events to application
 07h/03h    INFO-CONF   confirm receipt of INFO-IND
 08h/00h    DATA-REQ    send D-channel data
 08h/01h    DATA-CONF   confirm receipt of DATA-REQ
 08h/02h    DATA-IND    receive D-channel data
 08h/03h    DATA-RESP   confirm receipt fo DATA-IND
```

```
 09h/00h    CONNECT-INFO-REQ  request connection information
 09h/01h    CONNECT-INFO-CONF return connection information
 40h/00h    SELECT-B2-PROTOCOL-REQ  select Level 2 protocol
 40h/01h    SELECT-B2-PROTOCOL-CONF confirm receipt of SELECT-B2-PROTOCOL-REQ
 80h/00h    SELECT-B3-PROTOCOL-REQ  select Level 3 protocol
 80h/01h    SELECT-B3-PROTOCOL-CONF confirm receipt of SELECT-B3-PROTOCOL-REQ
 81h/00h    LISTEN-B3-REQ enable notification of incoming Level 3 calls
 81h/01h    LISTEN-B3-CONF  confirm receipt of LISTEN-B3-REQ
 82h/00h    CONNECT-B3-REQ  establish Level 3 connection
 82h/01h    CONNECT-B3-CONF confirm start of connection establishment
 82h/02h    CONNECT-B3-IND  indication of incoming Level 3 connection
 82h/03h    CONNECT-B3-RESP accept incoming connection
 83h/02h    CONNECT-B3-ACTIVE-IND  indication that Level 3 connection complete
 83h/03h    CONNECT-B3-ACTIVE-RESP confirm connection-complete indication
 84h/00h    DISCONNECT-B3-REQ request shutdown of Level 3 connection
 84h/01h    DISCONNECT-B3-CONF  confirm shutdown of Level 3 connection
 84h/02h    DISCONNECT-B3-IND indication that Level 3 is shutting down
 84h/03h    DISCONNECT-B3-RESP  confirm that application knows of shutdown
 85h/00h    GET-B3-PARAMS-REQ request Level 3 parameters
 85h/01h    GET-B3-PARAMS-CONF  return Level 3 parameters
 86h/00h    DATA-B3-REQ   send data on Level 3
 86h/01h    DATA-B3-CONF  confirm sending of Level 3 data
 86h/02h    DATA-B3-IND   indication of incoming Level 3 data
 86h/03h    DATA-B3-RESP  confirm receipt of Level 3 data
 87h/02h    HANDSET-IND   indication of Handset events
 87h/03h    HANDSET-RESP  confirm receipt of Handset event
 FFh/00h    MANUFACTURER-REQ  vendor-specific request
 FFh/01h    MANUFACTURER-CONF vendor-specific request confirmation
 FFh/02h    MANUFACTURER-IND  vendor-specific notification
 FFh/03h    MANUFACTURER-RESP vendor-specific notification confirmation
SeeAlso: #04071,#04073
----------F1--03-----------------------------
INT F1 - Common ISDN API (CAPI) v2.0 - "CAPI_PUT_MESSAGE"
  AL = 03h
  AH = CAPI version number * 10 (14h for v2.0)
  DX = application ID (see INT F1/AL=01h)
  ES:BX -> message to be sent (see #04071)
Return: AX = status (0000h,11xxh) (see #04070)
Notes:  the caller is required to provide at least 512 bytes of stack space
  the message buffer may be reused as soon as this call returns
SeeAlso: AH=03h,INT F1/AL=01h,INT F1/AL=04h
```

(Table 04073)
Values for CAPI v2.0 message command/subcommand:

Cmd/SubCmd	Name	Description
01h/80h	ALERT_REQ	indicate compatibility with incoming calls
01h/81h	ALERT_CONF	confirm receipt of ALERT_REQ
02h/80h	CONNECT_REQ	establish B-channel connection
02h/81h	CONNECT_CONF	confirm start of connection establishment
02h/82h	CONNECT_IND	indication from CAPI of incoming connection
02h/83h	CONNECT_RESP	accept incoming connection
03h/82h	CONNECT_ACTIVE_IND	indication that B-channel connection complete
03h/83h	CONNECT_ACTIVE_RESP	confirm connection-complete indication
04h/80h	DISCONNECT_REQ	request shutdown of B-channel connection
04h/81h	DISCONNECT_CONF	confirm shutdown of B-channel connection
04h/82h	DISCONNECT_IND	indication that B-channel is shutting down
04h/83h	DISCONNECT_RESP	confirm that application knows of shutdown
05h/80h	LISTEN_REQ	enable signalling on incoming events
05h/81h	LISTEN_CONF	confirm enabling of incoming event signalling
08h/80h	INFO_REQ	send protocol information for physical connect
08h/81h	INFO_CONF	confirm INFO_REQ
08h/82h	INFO_IND	indicate event for physical connection
08h/83h	INFO_CONF	confirm receipt of INFO_IND
41h/80h	SELECT_B_PROTOCOL_REQ	change protocol on already-active connect
41h/81h	SELECT_B_PROTOCOL_CONF	confirm receipt of SELECT_B_PROTOCOL_REQ
80h/80h	FACILITY_REQ	control optional facilities
80h/81h	FACILITY_CONF	confirm acceptance of FACILITY_REQ
80h/82h	FACILITY_IND	indicate facility-dependent event
80h/83h	FACILITY_RESP	confirm receipt of FACILITY_IND
82h/80h	CONNECT_B3_REQ	establish Level 3 connection
82h/81h	CONNECT_B3_CONF	confirm start of connection establishment
82h/82h	CONNECT_B3_IND	indication of incoming Level 3 connection
82h/83h	CONNECT_B3_RESP	accept incoming connection
83h/82h	CONNECT_B3_ACTIVE_IND	indication that Level 3 connection complete
83h/83h	CONNECT_B3_ACTIVE_RESP	confirm connection-complete indication
84h/80h	DISCONNECT_B3_REQ	request shutdown of Level 3 connection
84h/81h	DISCONNECT_B3_CONF	confirm shutdown of Level 3 connection
84h/82h	DISCONNECT_B3_IND	indication that Level 3 is shutting down
84h/83h	DISCONNECT_B3_RESP	confirm that application knows of shutdown
85h/80h	GET_B3_PARAMS_REQ	request Level 3 parameters
85h/81h	GET_B3_PARAMS_CONF	return Level 3 parameters
86h/80h	DATA_B3_REQ	send data on Level 3

```
 86h/81h    DATA_B3_CONF  confirm sending of Level 3 data
 86h/82h    DATA_B3_IND   indication of incoming Level 3 data
 86h/83h    DATA_B3_RESP  confirm receipt of Level 3 data
 87h/80h    RESET_B3_REQ  request resetting of a logical connection
 87h/81h    RESET_B3_CONF confirm logical connection reset
 87h/82h    RESET_B3_IND  indication from CAPI that logical conn. reset
 87h/83h    RESET_B3_RESP confirm receipt of RESET_B3_IND
 88h/82h    CONNECT_B3_T90_ACTIVE_IND  indicate switch from T.70 to T.90
 88h/83h    CONNECT_B3_T90_ACTIVE_RESP confirm receipt of T90_ACTIVE_IND
 FFh/80h    MANUFACTURER_REQ  vendor-specific request
 FFh/81h    MANUFACTURER_CONF vendor-specific request confirmation
 FFh/82h    MANUFACTURER_IND  vendor-specific notification
 FFh/83h    MANUFACTURER_RESP vendor-specific notification confirmation
SeeAlso: #04071,#04072
----------F104------------------------------
INT F1 - Common ISDN API (CAPI) v1.1 - "API-GET-MESSAGE"
  AH = 04h
  DX = application ID (see AH=01h)
Return: AX = status (0000h,1002h,1006h,1007h) (see #04069)
  ES:BX -> message if successful (see #04071)
Range:  INT 00 to INT FF, selectable by program parameter
Notes:  the caller is required to provide at least 512 bytes of stack space
  the returned message may be overwritten by the next call to this
    function
SeeAlso: AH=03h
----------F1--04------------------------------
INT F1 - Common ISDN API (CAPI) v2.0 - "CAPI_GET_MESSAGE"
  AL = 04h
  AH = CAPI version number * 10 (14h for v2.0)
  DX = application ID (see AH=01h)
Return: AX = status (0000h,11xxh) (see #04070)
  ES:BX -> message if successful (see #04071)
Range:  INT 00 to INT FF, selectable by program parameter
Notes:  the caller is required to provide at least 512 bytes of stack space
  the returned message may be overwritten by the next call to this
    function
SeeAlso: AH=04h,INT F1/AL=03h
----------F105------------------------------
INT F1 - Common ISDN API (CAPI) v1.1 - "API-SET-SIGNAL" - SIGNAL HANDLING
  AH = 05h
  DX = application ID (see AH=01h)
```

```
  ES:BX -> signal handler or 0000h:0000h to disable
Return: AX = status (0000h,1002h) (see #04069)
Notes:  the caller is required to provide at least 512 bytes of stack space
  the signal handler is called as though it were an interrupt, with
    interrupts disabled; the handler must preserve all registers and
    return with an IRET
SeeAlso: AH=01h
----------F1--05-----------------------------
INT F1 - Common ISDN API (CAPI) v2.0 - "CAPI_SET_SIGNAL" - SIGNAL HANDLING
  AL = 05h
  AH = CAPI version number * 10 (14h for v2.0)
  DX = application ID (see AH=01h)
  ES:BX -> signal handler or 0000h:0000h to disable
  SI:DI = parameter to pass to signal handler
Return: AX = status (0000h,11xxh) (see #04070)
Notes:  the caller is required to provide at least 512 bytes of stack space
  the signal handler is called as though it were an interrupt, with
    interrupts disabled and DX,SI,DI set as they were when this function
    was called; the handler must preserve all registers and return with
    an IRET
  the signal handler may call CAPI_PUT_MESSAGE, CAPI_GET_MESSAGE, and
    CAPI_SET_SIGNAL
SeeAlso: INT F1/AL=01h
----------F106-------------------------------
INT F1 - Common ISDN API (CAPI) v1.1 - "API-DEINSTALL" - UNINSTALL
  AH = 06h
  BX = force flag
       0000h normal uninstall
       0001h forced uninstall
Return: AX = status (0000h,1008h) (see #04069)
Desc: reset ISDN controller, close all ISDN Level 1 connections except for
    telephone connections
Notes:  the caller is required to provide at least 512 bytes of stack space
SeeAlso: INT F1/AL=01h,INT F1/AH=01h
----------F1F0-------------------------------
INT F1 - Common ISDN API (CAPI) v1.1 - "API-GET-MANUFACTURER"
  AH = F0h
  ES:BX -> 64-byte buffer for manufacturer identification information
Return: ES:BX buffer filled with ASCIZ idnetification string
Range:  INT 00 to INT FF, selectable by program parameter
Notes:  the caller is required to provide at least 512 bytes of stack space
```

```
SeeAlso: AH=01h,AH=F1h,AH=F2h,AH=FFh,INT F1/AL=F0h
----------F1--F0-----------------------------
INT F1 - Common ISDN API (CAPI) v2.0 - "CAPI_GET_MANUFACTURER"
  AL = F0h
  AH = CAPI version number * 10 (14h for v2.0)
  ES:BX -> 64-byte buffer for manufacturer identification information
Return: ES:BX buffer filled with ASCIZ idnetification string
Range:  INT 00 to INT FF, selectable by program parameter
Notes:  the caller is required to provide at least 512 bytes of stack space
SeeAlso: AH=F0h,INT F1/AL=01h,INT F1/AL=F1h,INT F1/AL=F2h,INT F1/AL=FFh
----------F1F1-----------------------------
INT F1 - Common ISDN API (CAPI) v1.1 - "API-GET-VERSION"
  AH = F1h
  ES:BX -> 64-byte buffer for CAPI version number
Return: ES:BX buffer filled with ASCIZ version string
Notes:  the caller is required to provide at least 512 bytes of stack space
SeeAlso: AH=01h,AH=F0h,AH=F2h,AH=FFh
----------F1--F1-----------------------------
INT F1 - Common ISDN API (CAPI) v2.0 - "CAPI_GET_VERSION"
  AL = F1h
  AH = CAPI version number * 10 (14h for v2.0)
Return: AH = CAPI major version number
  AL = CAPI minor version number
  DH = vendor-specific major version
  DL = vendor-specific minor version
Notes:  the caller is required to provide at least 512 bytes of stack space
SeeAlso: AH=F1h,INT F1/AL=01h,INT F1/AL=F0h,INT F1/AL=F2h,INT F1/AL=FFh
----------F1F2-----------------------------
INT F1 - Common ISDN API (CAPI) v1.1 - "API-GET-SERIAL-NUMBER"
  AH = F2h
  ES:BX -> 64-byte buffer for CAPI serial number
Return: ES:BX buffer filled with ASCIZ serial number (seven digits), empty
        string if no serial number
Notes:  the caller is required to provide at least 512 bytes of stack space
SeeAlso: AH=01h,AH=F0h,AH=F1h,AH=FFh
----------F1--F2-----------------------------
INT F1 - Common ISDN API (CAPI) v2.0 - "CAPI_GET_SERIAL_NUMBER"
  AL = F2h
  AH = CAPI version number * 10 (14h for v2.0)
  ES:BX -> 64-byte buffer for CAPI serial number
Return: ES:BX buffer filled with ASCIZ serial number (seven digits), empty
```

```
        string if no serial number
Notes:  the caller is required to provide at least 512 bytes of stack space
SeeAlso: AH=F2h,INT F1/AL=01h,INT F1/AL=F0h,INT F1/AL=F1h,INT F1/AL=F3h
----------F1--F3----------------------------
INT F1 - Common ISDN API (CAPI) v2.0 - "CAPI_GET_PROFILE" - GET CAPABILITIES
  AL = F3h
  AH = CAPI version number * 10 (14h for v2.0)
  ES:BX -> 64-byte buffer for CAPI capabilities (see #04074)
  CX = controller number (01h-06h) or 0000h to get number of controllers
Return: AX = status (0000h,11xxh) (see #04070)
  ES:BX buffer filled if successful
Notes:  the caller is required to provide at least 512 bytes of stack space
SeeAlso: INT F1/AL=01h,INT F1/AL=F0h,INT F1/AL=F2h,INT F1/AH=FFh

Format of CAPI v2.0 capabilities:
Offset  Size  Description (Table 04074)
 00h  WORD  number of installed controllers
 02h  WORD  number of supported B channels
 04h  DWORD global options (see #04075)
 08h  DWORD B1 protocol support flags (see #04076)
 0Ch  DWORD B2 protocol support flags (see #04077)
 10h  DWORD B3 protocol support flags (see #04078)
 14h 24 BYTEs reserved for CAPI use
 2Ch 20 BYTEs vendor-specific information

Bitfields for CAPI v2.0 global options:
Bit(s)  Description (Table 04075)
 0  internal controller supported
 1  external controller supported
 2  handset supported (only if bit 1 also set)
 3  DTMF supported
 4-31 reserved (0)
SeeAlso: #04074

Bitfields for CAPI v2.0 B1 protocol support:
Bit(s)  Description (Table 04076)
 0  64k bps with HDLC framing (required, always set)
 1  64k bps bit-transparent operation with network byte framing
 2  V.110 asynchronous with start/stop byte framing
 3  V.110 synchronous with HDLC framing
 4  T.30 modem for group 3 FAX
```

```
 5  64k bps inverted with HDLC framing
 6  56k bps bit-transparent operation with network byte framing
 7-31 reserved (0)
SeeAlso: #04074

Bitfields for CAPI v2.0 B2 protocol support:
Bit(s)  Description (Table 04077)
 0  ISO 7776 (X.75 SLP) (required, always set)
 1  transparent
 2  SDLC
 3  Q.921 LAPD (D-channel X.25)
 4  T.30 for group 3 FAX
 5  point-to-point protocol (PPP)
 6  transparent (ignoring B1 framing errors)
 7-31 reserved (0)
SeeAlso: #04074

Bitfields for CAPI v2.0 B3 protocol support:
Bit(s)  Description (Table 04078)
 0  transparent (required, always set)
 1  T.90NL with T.70NL compatibility
 2  ISO 8208 (X.25 DTE-DTE)
 3  X.25 DCE
 4  T.30 for group 3 FAX
 5-31 reserved (0)
SeeAlso: #04074
----------F1FF--------------------------------
INT F1 - Common ISDN API (CAPI) v1.1 - "API-MANUFACTURER" - VENDOR-SPECIFIC
  AH = FFh
  other registers vendor-specific
Return: registers vendor-specific
Range:  INT 00 to INT FF, selectable by program parameter
Notes:  the caller is required to provide at least 512 bytes of stack space
SeeAlso: AH=01h,AH=F0h,AH=F1h,AH=F2h,INT F1/AH=FFh
----------F1--FF-----------------------------
INT F1 - Common ISDN API (CAPI) v2.0 - "CAPI_MANUFACTURER" - VENDOR-SPECIFIC
  AL = FFh
  AH = CAPI version number * 10 (14h for v2.0)
  other registers vendor-specific
Return: registers vendor-specific
Range:  INT 00 to INT FF, selectable by program parameter
```

```
Notes:  the caller is required to provide at least 512 bytes of stack space
SeeAlso: AH=FFh,INT F1/AL=01h,INT F1/AL=F0h,INT F1/AL=F1h,INT F1/AL=F2h
--------*-F2--------------------------------
INT F2 - reserved for user interrupt
--------s-F2--------------------------------
INT F2 - Andy C. McGuire SPEECH.COM/SAY.COM
SeeAlso: INT F1"SPEECH"
--------N-F2--------------------------------
INT F2 - NetWare Remote Boot - ORIGINAL INT 13
SeeAlso: INT F1"NetWare",INT F3"NetWare",INT F4"NetWare"
----------F2--------------------------------
INT F2 - ICCTSR 1.0 - ImageCapture COLOR Developer's Kit - API
  AH = function number (see #04079)
  ???
Return: ???
Program: ImageCapture is a product of International Computers
SeeAlso: INT F3"ICCTSR"

(Table 04079)
Values for ImageCapture function:
 01h  power up
 02h  power down
 03h  set controls
 04h  capture image
 05h  display image
 06h  read file
 07h  write file
 08h  write array
 09h  read pixel
 0Ah  write pixel
 0Bh  check if VGA present
 0Ch  set video mode
 0Dh  check for keystroke
 0Eh  delay
--------*-F3--------------------------------
INT F3 - reserved for user interrupt
SeeAlso: INT F2"user",INT F4"user"
--------s-F3--------------------------------
INT F3 - SoundBlaster - POINTER TO ECHO VALUE
Note: this is not a vector, but a pointer to a DWORD containing the echo
    value selected with SET-ECHO.EXE
```

```
SeeAlso: INT 2F/AX=FBFBh/ES=0000h
----------F3--------------------------------
INT F3 - ICCTSR 1.0 - HANDSHAKE ID VECTOR
Program: ImageCapture is a product of International Computers
SeeAlso: INT F2"ICCTSR"
--------N-F3--------------------------------
INT F3 - NetWare Remote Boot - BOOT ROM'S INT 13 HANDLER
SeeAlso: INT F1"NetWare",INT F2"NetWare",INT F4"NetWare"
--------*-F4--------------------------------
INT F4 - reserved for user interrupt
SeeAlso: INT F3"user",INT F5"user"
--------T-F4--------------------------------
INT F4 - DoubleDOS - GIVE UP REST OF CURRENT CLOCK TICK AND ALL OF NEXT TICK
SeeAlso: INT 21/AH=EEh"DoubleDOS",INT F5"DoubleDOS",INT FE"DoubleDOS"
--------N-F4--------------------------------
INT F4 - NetWare Remote Boot - ???
SeeAlso: INT F1"NetWare",INT F2"NetWare",INT F3"NetWare"
--------*-F5--------------------------------
INT F5 - reserved for user interrupt
SeeAlso: INT F4"user",INT F6"user"
--------T-F5--------------------------------
INT F5 - DoubleDOS - ???
SeeAlso: INT F4"DoubleDOS",INT F6"DoubleDOS"
--------*-F6--------------------------------
INT F6 - reserved for user interrupt
SeeAlso: INT F5"user",INT F7"user"
--------T-F6--------------------------------
INT F6 - DoubleDOS - ???
SeeAlso: INT F5"DoubleDOS",INT F7"DoubleDOS"
--------*-F7--------------------------------
INT F7 - reserved for user interrupt
SeeAlso: INT F6"user"
--------T-F7--------------------------------
INT F7 - DoubleDOS - ???
SeeAlso: INT F6"DoubleDOS"
----------F700------------------------------
INT F7 - FSBBS 2.0 - CONFIGURATION RECORD
  AH = 00h
  AL = function
       00h get configuration record
    Return: DS:DX -> configuration record
```

```
      01h set configuration record
    Return: nothing
      02h get path for option
    DS:DX -> option name
    Return: DS:DX -> path
      03h determine whether configuration record set
    Return: AX = status
          0000h set
          0001h not yet set
      04h get link state
    Return: AX = state
          0000h unlinked
          0001h linked
Notes:  this information is preliminary and still subject to change
  all of the INT F7 calls for FSBBS are used for interprogram
    communication between the BBS kernel and the programs it spawns
SeeAlso: AH=01h
----------F701--------------------------------
INT F7 - FSBBS 2.0 - USER RECORD
  AH = 01h
  AL = function
      00h get user record for user currently online
    Return: DS:DX -> user record
      01h set user record
    DS:DX -> user record
    Return: nothing
SeeAlso: AH=00h,AH=02h
----------F702--------------------------------
INT F7 - FSBBS 2.0 - GET ACCOUNT NAME
  AH = 02h
Return: DS:DX -> 8-character blank-padded account name
SeeAlso: AH=01h
----------F703--------------------------------
INT F7 - FSBBS 2.0 - TERMINAL NUMBER
  AH = 03h
  AL = function
      00h get terminal index number
    Return: DX = index number
      01h set terminal index number
    DX = terminal index
    Return: nothing
```

```
----------F704--------------------------------
INT F7 - FSBBS 2.0 - PASSDATA BUFFER
  AH = 04h
  AL = function
       00h get PassData buffer contents
    DS:DX -> buffer for PassData contents
    Return: DS:DX buffer filled
       01h set PassData contents
    DS:DX -> buffer containing new PassData
    CH = length of data in buffer
    Return: nothing
----------F705--------------------------------
INT F7 - FSBBS 2.0 - TIMER FUNCTIONS
  AH = 05h
  AL = function
       00h get time remaining
    Return: DX = number of minutes remaining
       01h get current time
    Return: DS:DX -> 8-character time string
       02h increment time
    DX = number of additional minutes
       03h decrement time
    DX = number of minutes
SeeAlso: AH=06h
----------F706--------------------------------
INT F7 - FSBBS 2.0 - FUNCTION AVAILABILITY
  AH = 06h
  AL = function
       00h determine whether function is available
    DX = index of function
       01h set function availability
    DX = index of function
    ???
Return: nothing
SeeAlso: AH=05h,AH=07h
----------F707--------------------------------
INT F7 - FSBBS 2.0 - DUMP FUNCTIONS
  AH = 07h
  AL = function
       00h get current dump mode
    Return: DL = mode
```

```
      01h set dump mode
    DL = mode
SeeAlso: AH=06h
--------*-F8---------------------------------
INT F8 - reserved for user interrupt
--------h-F8---------------------------------
INT F8 - Sanyo MBC-550/555 - IRQ0 - 100 HZ INTERRUPT
Note: normally masked off at 8259 interrupt controller
SeeAlso: INT 08"IRQ0",INT F9"Sanyo",INT FA"Sanyo"
--------T-F8---------------------------------
INT F8 - DoubleDOS - ???
--------*-F9---------------------------------
INT F9 - reserved for user interrupt
--------T-F9---------------------------------
INT F9 - DoubleDOS - ???
--------h-F9---------------------------------
INT F9 - Sanyo MBC-550/555 - IRQ1 - ???
Note: documented as "for system use only"; normally enabled at the 8259
SeeAlso: INT 09"IRQ1",INT F8"Sanyo",INT FA"Sanyo"
--------*-FA---------------------------------
INT FA - reserved for user interrupt
--------h-FA---------------------------------
INT FA - Sanyo MBC-550/555 - IRQ2 - SERIAL PORT USART INTERRUPT
Note: this vector is not used on the Tandy 1000TL
SeeAlso: INT 0A"IRQ2",INT F9"Sanyo",INT FB"Sanyo"
--------T-FA---------------------------------
INT FA - DoubleDOS - TURN OFF TIMESHARING
SeeAlso: INT 21/AH=EAh"DoubleDOS",INT FB"DoubleDOS"
----------FA---------------------------------
INT FA P - ASM Edit - INSTALLATION CHECK
Program: ASM Edit is a shareware programmer's editor
Note: ASM Edit hooks this vector in protected mode to allow DPMI programs
    to detect whether they were run while shelled to DOS from ASM Edit

Format of ASM Edit signature block:
Offset  Size  Description (Table 04080)
 00h  BYTE  CFh (IRET)
 01h  8 BYTEs signature "ASM Edit" (no trailing NUL)
--------*-FB---------------------------------
INT FB - reserved for user interrupt
--------h-FB---------------------------------
```

```
INT FB - Sanyo MBC-550/555 - IRQ3 - KEYBOARD USART RECEIVE INTERRUPT
SeeAlso: INT 0B"IRQ3",INT FA"Sanyo",INT FC"Sanyo"
--------T-FB---------------------------------
INT FB - DoubleDOS - TURN ON TIMESHARING
SeeAlso: INT 21/AH=EBh"DoubleDOS",INT FA"DoubleDOS"
--------*-FC---------------------------------
INT FC - reserved for user interrupt
--------T-FC---------------------------------
INT FC - DoubleDOS - GET CURRENT SCREEN BUFFER ADDRESS
Return: ES = segment of display buffer
Desc: determine the address of the virtual screen to which the program
    should write instead of the actual video memory, so that the
    multitasked programs do not interfere with each other's output
Note: the display buffer may be moved if multitasking is enabled
SeeAlso: INT 21/AH=ECh"DoubleDOS",INT FB"DoubleDOS"
--------h-FC---------------------------------
INT FC - Sanyo MBC-550/555 - IRQ4 - PRINTER READY INTERRUPT
Note: normally masked off at the 8259 interrupt controller
SeeAlso: INT 0C"IRQ4",INT FB"Sanyo",INT FD"Sanyo"
--------*-FD---------------------------------
INT FD - reserved for user interrupt
--------T-FD---------------------------------
INT FD - DoubleDOS - ???
--------h-FD---------------------------------
INT FD - Sanyo MBC-550/555 - IRQ5 - FLOPPY DISK CONTROLLER
SeeAlso: INT 0D"IRQ5",INT FC"Sanyo",INT FE"Sanyo"
--------S-FD---------------------------------
INT FD - TFPCX - INSTALLATION CHECK
  AH = function (also see separate entries below)
Program: TFPCX is an interface between modem and terminal program for packet-
    radio communications
InstallCheck: test for the string "N5NX" three bytes beyond the interrupt
    handler
Range:  INT FD is the default, but may be changed, so the full installation
   check consists of scanning for the signature
Note: TFPCX returns AX=FFFFh on any unsupported function call
SeeAlso: AH=01h,AH=03h,AH=FEh
--------S-FD01-------------------------------
INT FD - TFPCX - TEST FOR CHARACTER WAITING
  AH = 01h
Return: AX = status
```

```
      0000h no characters waiting
      0001h character available for input
Program: TFPCX is an interface between modem and terminal program for packet-
    radio communications
SeeAlso: AH=02h
--------S-FD02-----------------------------
INT FD - TFPCX - GET CHARACTER
  AH = 02h
Return: AL = character
Notes:  this call is only allowed if AH=01h indicated that a character is
    available
  all available characters should be read before sending any additional
    characters
SeeAlso: AH=01h,AH=03h
--------S-FD03-----------------------------
INT FD - TFPCX - OUTPUT CHARACTER
  AH = 03h
  AL = character to send
Return: nothing
SeeAlso: AH=02h
--------S-FDFE-----------------------------
INT FD - TFPCX - GET VERSION
  AH = FEh
Return: AH = major version
  AL = minor version
Program: TFPCX is an interface between modem and terminal program for packet-
    radio communications
SeeAlso: AH=01h,AH=03h
--------B-FE-------------------------------
INT FE - AT/XT286/PS50+ - destroyed by return from protected mode
Note: the ROM BIOS uses 0030h:0100h as the initial stack on startup, which
    is the last fourth of the interrupt vector table.  If the processor
    is returned to real mode via a hardware reset (the only possibility
    on an 80286, though there are a number of ways of generating one),
    then the BIOS startup code stacks three words on its scratch stack
    before determining that a return to real mode has been requested.
    As a result, INT FE and INT FF are corrupted.
SeeAlso: INT FF"XT286"
--------T-FE-------------------------------
INT FE - DoubleDOS - GIVE UP TIME
  AL = number of 55ms time slices to give away
```

```
Return: after other program (if active) has run
SeeAlso: INT 21/AH=EEh"DoubleDOS",INT F4"DoubleDOS"
--------G-FE--------------------------------
INT FE - Turbo Debugger 8086 v2.5+ - OVERLAY MANAGER
SeeAlso: INT 3F
--------h-FE--------------------------------
INT FE - Sanyo MBC-550/555 - IRQ6 - 8087 COPROCESSOR INTERRUPT
Note: normally masked off at the 8259 interrupt controller
SeeAlso: INT 0E"IRQ6",INT FD"Sanyo",INT FF"Sanyo"
--------B-FF--------------------------------
INT FF - AT/XT286/PS50+ - destroyed by return from protected mode
Note: (see INT FE"XT286")
SeeAlso: INT FE"XT286"
--------b-FF--------------------------------
INT FF - Z100 - WARM BOOT
SeeAlso: INT 40"Z100"
--------h-FF--------------------------------
INT FF - Sanyo MBC-550/555 - IRQ7 - USER INTERRUPT FOR EXTERNAL INTERRUPT
Note: normally masked off at the 8259 interrupt controller
SeeAlso: INT 0F"IRQ7",INT FE"Sanyo"
--------Q-FF--------------------------------
INT FF U - QEMM-386.SYS v6.0+ - internal
Notes:  requires that a byte in the conventional-memory stub be set to the
    desired function number (00h through 0Ch)
SeeAlso: #04081

(Table 04081)
Values for QEMM internal functions:
 00h  reflect back to Virtual86-mode interrupt handler (default)
 01h  ???
 02h  access DR7???
 03h  QPI upcall (see INT 67/AH=3Fh)
 04h  ???
 05h  ???
 06h  INT 15/AH=87h
 07h  EMS services (see INT 67/AH=40h,INT 67/AH=5Dh)
 08h  ???
 09h  QEMM exception handler
 0Ah  XMS services (see INT 2F/AX=4310h"XMS")
 0Bh  Virtual DMA Services (see INT 4B/AX=8102h)
 0Ch  ???
```

```
--------v-FF---------------------------------
INT FF - VIRUS - "Violetta" - ???
Note: used but not chained by virus
SeeAlso: INT E0"VIRUS",INT F1"VIRUS"
--------V-FF----BX0000-----------------------
INT FF - PC/FORTH - GRAPHICS API - VIDEO STATUS CHANGE
  BX = 0000h
  DS:SI -> FORTH program counter
  SS:BP -> FORTH parameter stack
  SS:SP -> FORTH return stack
  DS:DX -> FORTH video parameter area
Desc: called to inform graphics driver of any status changes such as video
    mode changes, character color changes, graphics XOR mode turned on
    or off, etc.; also used as an installation check
Index:  installation check;PC/FORTH
--------V-FF---------------------------------
INT FF - PC/FORTH - GRAPHICS API
  BX = function number
       0001h function REDRAW
       0002h function !PEL
       0003h function @PEL
       0004h function LINE
       0005h function ARC
       0006h function @BLOCK
       0007h function !BLOCK
       0008h function FLOOD
  DS:SI -> FORTH program counter
  SS:BP -> FORTH parameter stack
  SS:SP -> FORTH return stack
  details of parameters not available
Return: AX,BX,CX,DX,ES,DI may be destroyed
Note: these functions all display an error message if the graphics routines
    are not resident
--------!---Admin----------------------------
Highest Table Number = 04122
--------!---FILELIST-------------------------
Please redistribute all of the files comprising the interrupt list (listed at
the beginning of the list and in INTERRUP.1ST) unmodified as a group, in a
quartet of archives named INTER61A through INTER61D (preferably the original
authenticated PKZIP archives), the utility programs in a fifth archive
called INTER61E.ZIP and the hypertext conversion programs in a sixth archive
```

named INTER61F.ZIP.

--------!---CONTACT_INFO--------------------

E-mail: ralf@pobox.com (currently forwards to ralf@telerama.lm.com)