Video Interrupts

int 10:BIOS video services:video interrupt
^INT 10 - Video BIOS Services

% For more information, see the following topics:

~INT 10,0~ - Set video mode
~INT 10,1~ - Set cursor type
~INT 10,2~ - Set cursor position
~INT 10,3~ - Read cursor position
~INT 10,4~ - Read light pen
~INT 10,5~ - Select active display page
~INT 10,6~ - Scroll active page up
~INT 10,7~ - Scroll active page down
~INT 10,8~ - Read character and attribute at cursor
~INT 10,9~ - Write character and attribute at cursor
~INT 10,A~ - Write character at current cursor
~INT 10,B~ - Set color palette
~INT 10,C~ - Write graphics pixel at coordinate
~INT 10,D~ - Read graphics pixel at coordinate
~INT 10,E~ - Write text in teletype mode
~INT 10,F~ - Get current video state
~INT 10,10~ - Set/get palette registers (EGA/VGA)
~INT 10,11~ - Character generator routine (EGA/VGA)
~INT 10,12~ - Video subsystem configuration (EGA/VGA)
~INT 10,13~ - Write string (BIOS after 1/10/86)
~INT 10,14~ - Load LCD char font (convertible)
~INT 10,15~ - Return physical display parms (convertible)
~INT 10,1A~ - Video Display Combination (VGA)
~INT 10,1B~ - Video BIOS Functionality/State Information (MCGA/VGA)
~INT 10,1C~ - Save/Restore Video State (VGA only)
~INT 10,FE~ - Get DESQView/TopView Virtual Screen Regen Buffer
~INT 10,FF~ - Update DESQView/TopView Virtual Screen Regen Buffer

Warning: Some BIOS implementations have a bug that causes register
BP to be destroyed. It is advisable to save BP before a call to
Video BIOS routines on these systems.

- registers CS, DS, ES, SS, BX, CX, DX are preserved unless
explicitly changed
- see ~INT 1F~ ~INT 1D~ ~INT 29~ ~INT 21,2~ ~INT 21,6~ ~INT 21,9~

:int 10,0:video modes
^INT 10,0 - Set Video Mode

AH = 00
AL = 00 40x25 B/W text (CGA,EGA,MCGA,VGA)
= 01 40x25 16 color text (CGA,EGA,MCGA,VGA)
= 02 80x25 16 shades of gray text (CGA,EGA,MCGA,VGA)

= 03 80x25 16 color text (CGA,EGA,MCGA,VGA)
= 04 320x200 4 color graphics (CGA,EGA,MCGA,VGA)
= 05 320x200 4 color graphics (CGA,EGA,MCGA,VGA)
= 06 640x200 B/W graphics (CGA,EGA,MCGA,VGA)
= 07 80x25 Monochrome text (MDA,HERC,EGA,VGA)
= 08 160x200 16 color graphics (PCjr)
= 09 320x200 16 color graphics (PCjr)
= 0A 640x200 4 color graphics (PCjr)
= 0B Reserved (EGA BIOS function 11)
= 0C Reserved (EGA BIOS function 11)
= 0D 320x200 16 color graphics (EGA,VGA)
= 0E 640x200 16 color graphics (EGA,VGA)
= 0F 640x350 Monochrome graphics (EGA,VGA)
= 10 640x350 16 color graphics (EGA or VGA with 128K)
640x350 4 color graphics (64K EGA)
= 11 640x480 B/W graphics (MCGA,VGA)
= 12 640x480 16 color graphics (VGA)
= 13 320x200 256 color graphics (MCGA,VGA)
= 8x EGA, MCGA or VGA ignore bit 7, see below
= 9x EGA, MCGA or VGA ignore bit 7, see below

- if AL bit 7=1, prevents EGA,MCGA & VGA from clearing display
- function updates byte at 40:49; bit 7 of byte 40:87
(EGA/VGA Display Data Area) is set to the value of AL bit 7

:int 10,1
^INT 10,1 - Set Cursor Type

AH = 01
CH = cursor starting scan line (cursor top) (low order 5 bits)
CL = cursor ending scan line (cursor bottom) (low order 5 bits)

returns nothing

- cursor scan lines are zero based
- cursor size can also be set via the ~6845~ CRT controller
- cursor size can be determined using the CRTC, ~INT 10,3~ or the
~BIOS Data Area~ bytes 40:60 (ending scan line) and 40:61 (starting
scan line)
- the following is a list of the cursor scan lines associated with
most common adapters; screen sizes over 40 lines may differ
depending on adapters.

% Video	Line Count	Starting Scan Line	Ending Scan Line	Character Point Size
CGA	25	06	07	08

MDA	25	0B	0C	0E
EGA	25	06	07	0E
EGA	43	04/06	07	08
VGA	25	0D	0E	10
VGA	40	08	09	0A
VGA	50	06	07	08

- use CX = 2000h to disable cursor

:int 10,2
^INT 10,2 - Set Cursor Position

AH = 02
BH = page number (0 for graphics modes)
DH = row
DL = column

returns nothing

- positions relative to 0,0 origin
- 80x25 uses coordinates 0,0 to 24,79; 40x25 uses 0,0 to 24,39
- the ~6845~ can also be used to perform this function
- setting the data in the BIOS Data Area at location 40:50 does not take immediate effect and is not recommended
- see ~VIDEO PAGES~ ~6845~ ~BDA~

:int 10,3
^INT 10,3 - Read Cursor Position and Size

AH = 03
BH = video page

on return:
CH = cursor starting scan line (low order 5 bits)
CL = cursor ending scan line (low order 5 bits)
DH = row
DL = column

- returns data from ~BIOS DATA AREA~ locations 40:50, 40:60 and 40:61
- the ~6845~ can also be used to read the cursor position
- the return data can be circumvented by direct port I/O to the 6845 CRT Controller since this function returns the data found in the BIOS Data Area without actually checking the controller

:int 10,4

^INT 10,4 - Read Light Pen Position

AH = 04

on return:
AH = 0 light pen switch not triggered
  = 1 light pen triggered
BX = pixel column (0-319 or 0-639, mode dependent)
CH = raster line (0-199) (CGA and EGA modes 4, 5 and 6)
CX = raster line (EGA modes except 4, 5 and 6)
DH = row (0-24)
DL = column (0-79 or 0-79 mode dependent)

- data returned as a byte coordinate, leaving horizontal
  accuracy to within 2 pixels (320) or 4 pixels (640)
- vertical accuracy within 2 lines
- PS/2's don't support the light pen interface

:int 10,5
^INT 10,5 - Select Active Display Page

AH = 05
AL = new page number, see ~VIDEO PAGES~

for PCjr only:
AL = 80h to read CRT/CPU page registers
  81h to set CPU page register
    BL = CPU page register
  82h to set CRT page register
    BH = CRT page register
  83h to set CPU and page registers
    BH = CRT page register
    BL = CPU page register

on return: (PCjr only)
BH = CRT page register
BL = CPU page register

:int 10,6
^INT 10,6 - Scroll Window Up

AH = 06
AL = number of lines to scroll, previous lines are
  blanked, if 0 or AL > screen size, window is blanked
BH = attribute to be used on blank line
CH = row of upper left corner of scroll window

CL = column of upper left corner of scroll window
DH = row of lower right corner of scroll window
DL = column of lower right corner of scroll window

returns nothing

- in video mode 4 (300x200 4 color) on the EGA, MCGA and VGA
  this function scrolls page 0 regardless of the current page
- can be used to scroll graphics screens, using character coords
- on CGA's this function disables video adapter, causing flitter

:int 10,7
^INT 10,7 - Scroll Window Down

AH = 07
AL = number of lines to scroll, previous lines are
    blanked, if 0 or AL > screen size, window is blanked
BH = attribute to be used on blank line
CH = row of upper left corner of scroll window
CL = column of upper left corner of scroll window
DH = row of lower right corner of scroll window
DL = column of lower right corner of scroll window

returns nothing

- in video mode 4 (300x200 4 color) on the EGA, MCGA and VGA
  this function scrolls page 0 regardless of the current page
- can be used to scroll graphics screens, using character coords
- on CGA's this function disables video adapter, causing flitter

:int 10,8
^INT 10,8 - Read Character and Attribute at Cursor Position

AH = 08
BH = display page

on return:
AH = attribute of character (alpha modes only)
AL = character at cursor position

- in video mode 4 (300x200 4 color) on the EGA, MCGA and VGA
  this function works only on page zero

:int 10,9
^INT 10,9 - Write Character and Attribute at Cursor Position

AH = 09
AL = ASCII character to write
BH = display page (or mode 13h, background pixel value)
BL = character attribute (text) foreground color (graphics)
CX = count of characters to write (CX >= 1)

returns nothing

- does not move the cursor
- in graphics mode (except mode 13h), if BL bit 7=1 then
  value of BL is XOR'ed with the background color

:int 10,a
^INT 10,A - Write Character Only at Current Cursor Position

AH = 0A
AL = ASCII character to write
BH = display page (or mode 13h, background pixel value)
BL = foreground color (graphics mode only)
CX = count of characters to write (CX >= 1)

return nothing

- similar to ~INT 10,9~ except color ignored in text modes

:int 10,b
^INT 10,B - Set Color Palette

AH = 0B
BH = palette color ID
  = 0 to set background and border color
  = 1 to select 4 color palette
BL = color value (when BH = 0)
  = palette value (when BH = 1)

Palette	Pixel	Color
0	0	current background color
	1	green (2)
	2	red (4)
	3	brown (6)
1	0	current background color
	1	cyan (3)

2 magenta (5)
3 white (7)

- does not work for all EGA and VGA video modes
- sets border color in text mode (BH = 0)

:int 10,c
^INT 10,C - Write Graphics Pixel at Coordinate

AH = 0C
AL = color value (XOR'ED with current pixel if bit 7=1)
BH = page number, see ~VIDEO PAGES~
CX = column number (zero based)
DX = row number (zero based)

returns nothing

- if bit 7 is 1, color specified is XOR'ed with current pixel
- page number in BH ignored for 320x200 4 color graphics mode
- this function is known to destroy AX and possibly SI and DI on on some PS/2 VGA systems

:int 10,d
^INT 10,D - Read Graphics Pixel at Coordinate

AH = 0D
BH = page number, see ~VIDEO PAGES~
CX = column number (zero based)
DX = row number (zero based)

on return:
AL = color of pixel read

- 64K IBM EGAs with BIOS dated 9/13/84 in 350 line video modes, return invalid data in AL
- page number in BH ignored for 320x200 4 color graphics mode

:int 10,e
^INT 10,E - Write Text in Teletype Mode

AH = 0E
AL = ASCII character to write
BH = page number (text modes)

BL = foreground pixel color (graphics modes)

returns nothing

- cursor advances after write
- characters BEL (7), BS (8), LF (A), and CR (D) are
  treated as control codes
- for some older BIOS (10/19/81), the BH register must point
  to the currently displayed page
- on CGA adapters this function can disable the video signal while
  performing the output which causes flitter.

:int 10,f
^INT 10,F - Get Video State

AH = 0F

on return:
AH = number of screen columns
AL = mode currently set (see ~VIDEO MODES~)
BH = current display page

- video modes greater than 13h on EGA, MCGA and VGA indicate
  ~INT 10,0~ was called with the high bit of the mode (AL) set
  to 1, meaning the display does not need cleared
- function returns byte value at 40:49; On EGA, MCGA and
  VGA bit 7 of register AL is determined by bit 7 of BIOS Data
  Area byte 40:87. This bit is usually set by INT 10,0
  with bit 7 of the requested mode (in AL) set to 1

:int 10,10
^INT 10,10 - Set/Get Palette Registers (EGA/VGA)

AH = 10h

% AL = 00 set individual palette register
  BH = color value
  BL = palette register

% AL = 01 set border color (overscan register)
  BH = color value

% AL = 02 set all palette registers and border

ES:DX = pointer to 17 byte table representing 16 palette
registers and border color register

% AL = 03 toggle intensity/blinking (EGA)
BL = 0 enable intensity
1 enable blinking

% AL = 07 read palette register (PS/2)
BL = palette register to read (0-15)

on return:
BH = value of palette register

% AL = 08 read border color (overscan register, PS/2)

on return:
BH = value of border color (overscan register)

% AL = 09 read palette registers and border (PS/2)
ES:DX = pointer to 17 byte table representing 16 palette
registers and border color register

on return:
ES:DX = pointer to table provided as input

% AL = 10 set DAC color register
BX = color register to set
CH = green value
CL = blue value
DH = red value

% AL = 12 set block of DAC color registers
BX = first color register to set
CX = number of color registers to set
ES:DX = pointer to table of color values to set

% AL = 13 set attribute controller color select state
BL = 0 set Mode Control register bit 7
BH = value for bit 7
BL = 1 set color select register
BH = value for color select register

% AL = 15 read DAC color register (PS/2)
   BX = color register to read

on return:
CH = green value
CL = blue value
DH = red value

% AL = 17 read block of DAC color registers
   BX = first color register to read
   CX = number of color registers to read
   ES:DX = pointer to buffer for color registers

on return:
ES:DX = pointer to color table provided as input

% AL = 18 update video DAC mask register
   BL = new mask

% AL = 19 read video DAC mask register

on return:
BL = value read from video DAC mask register

% AL = 1A read color page state
   BL = bit 7 of Mode Control Register
   BH = bits 2 thru 3 of Color select register if BL = 0
      = bits 0 thru 3 of Color select register if BL = 1

on return:
BL = current paging mode
CX = current page

% AL = 1B sum color values to shades of gray
   BX = first color register to sum
   CX = number of color registers to sum

- controls the pixel color mapping bit values
- BIOS extension to EGA/VGA systems

:int 10,11
^INT 10,11 - Character Generator Routine (EGA/VGA)

AH = 11h

% AL = 00 user character load
BH = number of bytes per character
BL = table in character generator RAM
CX = count of characters in table
DX = ASCII code of first character defined
ES:BP = pointer to user table

% AL = 01 ROM BIOS 8x14 monochrome set
BL = table in character generator RAM

% AL = 02 ROM BIOS 8x8 double dot
BL = table in character generator RAM

% AL = 03 set displayed definition table
BL = value for character Map Select register (EGA,VGA)
= character generator RAM table numbers (MCGA)

% AL = 04 ROM BIOS 8x16 character set
BL = table in character generator RAM

% AL = 10 user specified character definition table
BH = bytes per character (points)
BL = table in character generator RAM
CX = number of characters defined in table
DX = ASCII code of first character defined
ES:BP = pointer to user table

% AL = 11 ROM BIOS 8x14 monochrome character set
BL = table in character generator RAM

% AL = 12 ROM 8x8 double dot character definitions
BL = table in character generator RAM

% AL = 14 ROM 8x16 double dot character definitions
BL = table in character generator RAM

% AL = 20 pointer to graphics character table for ~INT 1F~ (8x8)
ES:BP = pointer to user table

% AL = 21 user graphics character pointer at INT 43
BL = row specifier
= 0 - user specified (DL = rows)
= 1 is 14 rows
= 2 is 25 rows
= 3 is 43 rows
CX = bytes per character (points)
DL = rows (when BL = 0)
ES:BP = pointer to user table

% AL = 22 ROM 8x14 character set
BL = number of rows (see AL=21)
DL = rows (when BL = 0)

% AL = 23 ROM 8x8 double dot character set
BL = row specifier (see AL=21)
DL = rows (when BL = 0)

% AL = 24 ROM 8x16 character set
BL = row specifier (see AL=21)
DL = rows (when BL = 0)

% AL = 30 get current character generator information
BH = information desired:
= 0 ~INT 1F~ pointer
= 1 INT 44h pointer
= 2 ROM 8x14 pointer
= 3 ROM 8x8 double dot pointer (base)
= 4 ROM 8x8 double dot pointer (top)
= 5 ROM 9x14 alpha alternate pointer
= 6 ROM 8x16 character table pointer
= 7 ROM 9x16 alternate character table pointer

on return:
CX = bytes per character (points)
DL = rows (less 1)
ES:BP = pointer to table

:int 10,12
^INT 10,12 - Video Subsystem Configuration (EGA/VGA)

AH = 12h

% BL = 10 return video configuration information

on return:
BH = 0 if color mode in effect
  = 1 if mono mode in effect
BL = 0 if 64k EGA memory
  = 1 if 128k EGA memory
  = 2 if 192k EGA memory
  = 3 if 256k EGA memory
CH = feature bits
CL = switch settings

% BL = 20 select alternate print screen routine

% BL = 30 select scan lines for alphanumeric modes
  AL = 0 200 scan lines
    = 1 350 scan lines
    = 2 400 scan lines

on return:
AL = 12

% BL = 31 select default palette loading
  AL = 0 enable default palette loading
    = 1 disable default palette loading

on return:
AL = 12

% BL = 32 CPU access to video RAM
  AL = 0 enable CPU access to video RAM and I/O ports
    = 1 disable CPU access to video RAM and I/O ports

on return:
AL = 12

% BL = 33 Gray scale summing
  AL = 0 enable gray scale summing
    = 2 disable gray scale summing

on return:
AL = 12

% BL = 34 cursor emulation
  AL = 0 enable cursor emulation
    = 1 disable cursor emulation

on return:

AL = 12

% BL = 35 PS/2 video display switching
AL = 0 initial adapter video off
= 1 initial planar video on
= 2 switch active video off
= 3 switch inactive video on
ES:DX pointer to 128 byte save area (when AL = 0, 2 or 3)

on return:
AL = 12

% BL = 36 video refresh control
AL = 0 enable refresh
= 1 disable refresh

on return:
AL = 12

:int 10,13
^INT 10,13 - Write String (BIOS versions from 1/10/86)

AH = 13h
AL = write mode (see bit settings below)
= 0 string is chars only, attribute in BL, cursor not moved
= 1 string is chard only, attribute in BL, cursor moved
= 2 string contains chars and attributes, cursor not moved
= 3 string contains chars and attributes, cursor moved
BH = video page number
BL = attribute if mode 0 or 1 (AL bit 1=0)
CX = length of string (ignoring attributes)
DH = row coordinate
DL = column coordinate
ES:BP = pointer to string

Bit settings for write mode (register AL):

³7³6³5³4³3³2³1³0³ AL
³ ³ ³ ³ ³ ³ ÀÄÄÄÄ 0=don't move cursor, 1=move cursor
³ ³ ³ ³ ³ ³ ÀÄÄÄÄÄ 0=BL has attributes, 1=string has attributes
ÀÄÁÄÄÁÄÄÁÄÄÁÄÄÄÄÄÄ unused

returns nothing

- BEL, BS, CR, LF are treated as ASCII control codes
- wraps data and scrolls if unable to fit data on one line

:int 10,14
^INT 10,14 - Load LCD Character Font (convertible only)

AH = 14h

% AL = 0 - load user specified font
ES:DI = pointer to character font
CX = number of characters to store
DX = char offset into ram font area
BH = number of bytes per character
BL = 0 load main font (block 0)
= 1 load alternate font (block 1)

% AL = 1 - load system ROM default font
BL = 0 load main font (block 0)
= 1 load alternate font (block 1)

% AL = 2 - set mapping of LCD high intensity attribute
BL = 0 ignore high intensity attribute
= 1 map high intensity to underscore
= 2 map high intensity to reverse video
= 3 map high intensity to select alternate font

:int 10,15
^INT 10,15 - Return Physical Display Parms (convertible)

AH = 15h

on return:
AX = alternate display adapter type
ES:DI = pointer to parameter table:

% Offset Size Description

01 word monitor model number
02 word vertical pels per meter
03 word horizontal pels per meter
04 word total number of vertical pels
05 word total number of horizontal pels
06 word horizontal pel separation in micrometers
07 word vertical pel separation in micrometers

:int 10,1a
^INT 10,1A - Video Display Combination (VGA)

AH = 1A
AL = 00 get video display combination
= 01 set video display combination
BL = active display (see table below)

BH = inactive display

on return:
AL = 1A, if a valid function was requested in AH
BL = active display (AL=00, see table below)
BH = inactive display (AL=00)

% Valid display codes:

FF Unrecognized video system
00 No display
01 MDA with monochrome display
02 CGA with color display
03 Reserved
04 EGA with color display
05 EGA with monochrome display
06 Professional graphics controller
07 VGA with analog monochrome display
08 VGA with analog color display
09 Reserved
0A MCGA with digital color display
0B MCGA with analog monochrome display
0C MCGA with analog color display

- returns value at byte 40:8A indicating display combination status
- used to detect video display capabilities

:int 10,1b
^INT 10,1B - Video BIOS Functionality and
^State Information (MCGA/VGA)

AH = 1B
BX = implementation type (must be zero)
ES:DI = pointer to 64 byte buffer

on return:
AL = 1B
ES:DI = pointer to updated buffer (see below)

- returns static and dynamic information about the current
  state and capabilities of the current video system
- bytes 0-3 of the dynamic data table at ES:DI contain a far
  pointer to the video static information table

^Video BIOS Dynamic Functionality State Table (MCGA/VGA)

% Dynamic Video State Table

00 dword address of static functionality table
04 byte video mode
05 word number of columns
07 word length of displayed video buffer (# bytes)
09 word start address of upper left corner of video buffer
0B 16bytes cursor position table for 8 pages (col,row)
1B byte cursor end line
1C byte cursor start line
1D byte active video page
1E word I/O port for CRTC address register
20 byte current value of CRTC 3x8 register
21 byte current value of CRTC 3x9 register
22 byte number of displayed character rows
23 word height of character matrix (points)
25 byte active display combination code
26 byte inactive display combination code
27 word number of displayed colors (mono = 0)
29 byte number of supported video pages
2A byte raster scan lines 0=200, 1=350, 2=400, 3=480
2B byte text character table used
2C byte text character table used
2D byte other state information:

³7³6³5³4³3³2³1³0³ State Information byte at offset 2D
³ ³ ³ ³ ³ ³ ³ ÀÄÄÄ 1 = all modes active (MCGA always 0)
³ ³ ³ ³ ³ ³ ÀÄÄÄÄ 1 = gray scale summing enabled
³ ³ ³ ³ ³ ÀÄÄÄÄÄ 1 = monochrome display attached
³ ³ ³ ³ ÀÄÄÄÄÄÄ 1 = default palette loading disabled
³ ³ ³ ÀÄÄÄÄÄÄÄ 1 = cursor emulation enabled
³ ³ ÀÄÄÄÄÄÄÄÄ 1 = blinking attribute enabled
ÀÄÁÄÄÄÄÄÄÄÄ 1 = reserved

2E 3bytes reserved
31 byte video RAM available 0=64K, 1=128K, 2=192K, 3=256K
32 byte save area status

³7³6³5³4³3³2³1³0³ Save Area Status
³ ³ ³ ³ ³ ³ ³ ÀÄÄÄ 1 = two text char sets are active
³ ³ ³ ³ ³ ³ ÀÄÄÄÄ 1 = dynamic save area is active
³ ³ ³ ³ ³ ÀÄÄÄÄÄ 1 = text char set override is active
³ ³ ³ ³ ÀÄÄÄÄÄÄ 1 = graphics char set is override active
³ ³ ³ ÀÄÄÄÄÄÄÄ 1 = palette override is active
³ ³ ÀÄÄÄÄÄÄÄÄ 1 = display combination code ext. active
ÀÄÁÄÄÄÄÄÄÄÄ 1 = reserved

33 dword reserved

^Video BIOS Static Functionality Table (EGA/VGA)

³7³6³5³4³3³2³1³0³ Video modes supported, byte at offset 00

³ ³ ³ ³ ³ ³ ³ ÀÄÄÄ 1 = mode 0
³ ³ ³ ³ ³ ³ ÀÄÄÄÄ 1 = mode 1
³ ³ ³ ³ ³ ÀÄÄÄÄÄ 1 = mode 2
³ ³ ³ ³ ÀÄÄÄÄÄÄ 1 = mode 3
³ ³ ³ ÀÄÄÄÄÄÄÄ 1 = mode 4
³ ³ ÀÄÄÄÄÄÄÄÄ 1 = mode 5
³ ÀÄÄÄÄÄÄÄÄÄ 1 = mode 6
ÀÄÄÄÄÄÄÄÄÄÄ 1 = mode 7

³7³6³5³4³3³2³1³0³ Video modes supported, byte at offset 01
³ ³ ³ ³ ³ ³ ³ ÀÄÄÄ 1 = mode 8
³ ³ ³ ³ ³ ³ ÀÄÄÄÄ 1 = mode 9
³ ³ ³ ³ ³ ÀÄÄÄÄÄ 1 = mode A
³ ³ ³ ³ ÀÄÄÄÄÄÄ 1 = mode B
³ ³ ³ ÀÄÄÄÄÄÄÄ 1 = mode C
³ ³ ÀÄÄÄÄÄÄÄÄ 1 = mode D
³ ÀÄÄÄÄÄÄÄÄÄ 1 = mode E
ÀÄÄÄÄÄÄÄÄÄÄ 1 = mode F

³7³6³5³4³3³2³1³0³ Video modes supported, byte at offset 02
³ ³ ³ ³ ³ ³ ³ ÀÄÄÄ 1 = mode 10
³ ³ ³ ³ ³ ³ ÀÄÄÄÄ 1 = mode 11
³ ³ ³ ³ ³ ÀÄÄÄÄÄ 1 = mode 12
³ ³ ³ ³ ÀÄÄÄÄÄÄ 1 = mode 13
ÀÄÁÄÁÄÁÄÄÄÄÄÄÄ reserved

03 dword reserved
07 byte scan lines supported in text modes

³7³6³5³4³3³2³1³0³ Scan lines supported, byte at offset 07
³ ³ ³ ³ ³ ³ ³ ÀÄÄÄ 1 = 200 lines
³ ³ ³ ³ ³ ³ ÀÄÄÄÄ 1 = 350 lines
ÀÄÁÄÁÄÁÄÁÄÁÄÄÄÄÄ 1 = 400 lines

08 byte max number of displayable text character sets
09 byte # of text definition tables in char generator RAM
0A byte other capability flags

³7³6³5³4³3³2³1³0³ Other flags, byte at offset 0A
³ ³ ³ ³ ³ ³ ³ ÀÄÄÄ 1 = all modes (0 on MCGA)
³ ³ ³ ³ ³ ³ ÀÄÄÄÄ 1 = gray scale summing
³ ³ ³ ³ ³ ÀÄÄÄÄÄ 1 = character set loading
³ ³ ³ ³ ÀÄÄÄÄÄÄ 1 = default palette loading
³ ³ ³ ÀÄÄÄÄÄÄÄ 1 = cursor emulation
³ ³ ÀÄÄÄÄÄÄÄÄ 1 = 64 color palette
³ ÀÄÄÄÄÄÄÄÄÄ 1 = video DAC loading
ÀÄÄÄÄÄÄÄÄÄÄ 1 = DAC controlled by ACCS

0B byte other capability flags

³7³6³5³4³3³2³1³0³ Other flags, byte at offset 0B
³ ³ ³ ³ ³ ³ ³ ÀÄÄÄ 1 = light pen support

³ ³ ³ ³ ³ ³ ÀÄÄÄÄ 1 = save/restore video state
³ ³ ³ ³ ³ ÀÄÄÄÄÄ 1 = blinking/background intensity
³ ³ ³ ³ ÀÄÄÄÄÄÄ 1 = display combination code
ÀÄÁÄÁÄÁÄÄÄÄÄÄÄ reserved

0C word reserved
0E byte save area capabilities

³7³6³5³4³3³2³1³0³ save area capabilities at offset 0E
³ ³ ³ ³ ³ ³ ³ ÀÄÄÄÄ 1 = multiple text character sets
³ ³ ³ ³ ³ ³ ÀÄÄÄÄÄ 1 = dynamic save area
³ ³ ³ ³ ³ ÀÄÄÄÄÄÄ 1 = text character set override
³ ³ ³ ³ ÀÄÄÄÄÄÄÄ 1 = graphics character set override
³ ³ ³ ÀÄÄÄÄÄÄÄÄ 1 = palette override
³ ³ ÀÄÄÄÄÄÄÄÄÄ 1 = display combination code extension
ÀÄÁÄÄÄÄÄÄÄÄÄ reserved

0F byte reserved

:int 10,1c
^INT 10,1C - Save/Restore Video State (VGA only)

AH = 1C

% AL = 0 get save buffer size
CX = requested states
bit 0: video hardware state
bit 1: video BIOS data areas
bit 2: video DAC state

on return:
AL = 1C
BX = buffer size in 64 byte blocks

% AL = 1 save requested state
CX = requested states (see AL = 0)
ES:BX = pointer to buffer

returns nothing

% AL = 2 restore requested states
CX = requested states (see AL = 0)
ES:BX = pointer to buffer

returns nothing

:int 10,fe
^INT 10,FE - Get DESQView/TopView Virtual Screen Regen Buffer

AH = FE
ES:DI = set to sentinel value (test for INT 10,FE supported)

returns:
ES:DI = address of DESQView/TopView video buffer, DI will always
be zero

- on return ES:DI should be tested against the original value;
the value will change if this function is supported (DESQView
or TopView loaded), otherwise it will remain unchanged
- if ES:DI changes this address can be used as the video screen
regen buffer

:int 10,ff
^INT 10,FF - Update DESQView/TopView Virtual Screen Regen Buffer

AH = FF
CX = number of characters changed
ES:DI = pointer to first character in buffer to change, ES is
set to segment returned by ~INT 10,FE~

returns nothing

- the physical screen does not get updated until INT 10,FF is
called in TopView
- it is not necessary to make this call under DESQView since it
handles updates automatically
- calling this function under DESQView will cancel the automatic
update mode