

```

;*****
;*                               V I O S C A                               *
;*-----*
;* Task       : Creates a function for determining video adapter and monitor type, when linked with a C program. *
;*-----*
;* Author      : Michael Tischer *
;* Developed on : 10/02/88 *
;* Last update  : 02/18/92 *
;*-----*
;* Assembly    : MASM VIOSCA; *
;*              ... link to a C program *
;*****

;== Constants for VIOS structure =====

NO_VIOS    = 0           ;Video card constants
VGA        = 1           ;No video card
EGA        = 2           ;VGA card
MDA        = 3           ;EGA card
HGC        = 4           ;Monochrome Display Adapter
CGA        = 5           ;Hercules Graphics Card
                        ;Color Graphics Adapter

NO_MON     = 0           ;Monitor constants
MONO       = 1           ;No monitor
COLOR      = 2           ;Monochrome monitor
EGA_HIRES  = 3           ;Color monitor
ANLG_MONO  = 4           ;High-resolution or multisync monitor
ANLG_COLOR = 5           ;Analog monochrome monitor
                        ;Analog color monitor

;== Segment declarations for the C program =====

IGROUP group _text          ;Addition to program segment
DGROUP group const,_bss, _data ;Addition to data segment
      assume CS:IGROUP, DS:DGROUP, ES:DGROUP, SS:DGROUP

CONST segment word public 'CONST' ;This segment includes all read-only
CONST ends                       ;constants

_BSS segment word public 'BSS'    ;This segment includes all
_BSS ends                       ;un-initialized static variables

_DATA segment word public 'DATA' ;Data segment

vios_tab equ this byte

      ;-- Conversion table for return values of function 1AH, ----
      ;-- sub-function 00H of the VGA-BIOS ----

      db NO_VIOS, NO_MON      ;No video card
      db MDA, MONO           ;MDA card and monochrome monitor
      db CGA, COLOR          ;CGA card and color monitor
      db ?, ?                ;Code 3 unused
      db EGA, EGA_HIRES      ;EGA card and hi-res monitor
      db EGA, MONO           ;EGA card and monochrome monitor
      db ?, ?                ;Code 6 unused
      db VGA, ANLG_MONO      ;VGA card and analog mono monitor
      db VGA, ANLG_COLOR     ;VGA card and analog color monitor

ega_dips equ this byte

      ;-- Conversion table for EGA card DIP switch settings -----

      db COLOR, EGA_HIRES, MONO
      db COLOR, EGA_HIRES, MONO

_DATA ends

;== Program =====

_TEXT segment byte public 'CODE' ;Program segment

public _get_vios

```

```

;-----
;-- GET_VIOS: Determines types of installed video cards -----
;-- Call from C : void get_vios( struct vios *vp );
;-- Declaration : struct vios { BYTE vcard, monitor; };
;-- Return value: none
;-- Info      : This example uses function in SMALL memory model

_get_vios  proc near

sframe     struc                ;Stack access structure
cga_poss1  db ?                 ;Local variable
ega_poss1  db ?                 ;Local variable
mono_poss1 db ?                 ;Local variable
bptr       dw ?                 ;Take BP
ret_adr     dw ?                 ;Return address to caller
vp          dw ?                 ;Pointer to first VIOS structure
sframe     ends                 ;End of structure

frame      equ [ bp - cga_poss1 ] ;Address elements of the structure

        push bp                 ;Push BP onto stack
        sub  sp,3                ;Allocate space for local variables
        mov  bp,sp              ;Transfer SP to BP
        push di                 ;Push DI onto stack

        mov  frame.cga_poss1,1 ;Could be CGA
        mov  frame.ega_poss1,1 ;Could be EGA
        mov  frame.mono_poss1,1;Could be MDA or HGC

        mov  di,frame.vp        ;Get offset address of structure
        mov  word ptr [di],NO_VIOS ;Still no video
        mov  word ptr [di+2],NO_VIOS ;system found

        call test_vga           ;Test for VGA card
        cmp  frame.ega_poss1,0 ;EGA card still possible?
        je   gv1                ;No --> Test for CGA

gv1:     call test_ega           ;Test for EGA card
        cmp  frame.cga_poss1,0 ;CGA card still possible
        je   gv2                ;No --> Test for MDA/HGC

gv2:     call test_cga           ;Test for CGA card
        cmp  frame.mono_poss1,0;MDA or HGC card still possible?
        je   gv3                ;No --> End tests

        call test_mono          ;Test for MDA/HGC cards

;-- Determine active video card -----

gv3:     cmp  byte ptr [di],VGA ;VGA card active?
        je   gvi_end            ;Yes --> Active card determined
        cmp  byte ptr [di+2],VGA ;VGA card as secondary system?
        je   gvi_end            ;Yes --> Active card determined

        mov  ah,0Fh             ;Determine active video mode using
        int  10h                ;BIOS video interrupt

        and  al,7               ;Only modes 0-7 are of interest
        cmp  al,7               ;Monochrome card active?
        jne  gv4                ;NO, in CGA or EGA mode

;-- MDA, HGC, or EGA card (mono) is active -----

        cmp  byte ptr [di+1],MONO ;Mono monitor in first structure?
        je   gvi_end            ;Yes --> Sequence O.K.
        jmp  short switch       ;NO, Change sequence

;-- CGA or EGA card currently active -----

gv4:     cmp  byte ptr [di+1],MONO ;Mono monitor in first structure?
        jne  gvi_end            ;No --> Sequence O.K.

switch:  mov  ax,[di]            ;Get contents of first structure
        xchg ax,[di+2]          ;Exchange with second structure
        mov  [di],ax

```

```

gvi_end:    pop    di                ;Get DI from stack
            add    sp,3              ;Get local variables from stack
            pop    bp                ;Get BP from stack
            ret                     ;Return to C program

_get_vios   endp

;-----
;-- TEST_VGA: Determines whether a VGA card is installed

test_vga    proc near

            mov     ax,1a00h          ;Function 1AH, sub-function 00H
            int     10h               ;calls VGA-BIOS
            cmp     al,lah             ;Is this function supported?
            jne     tvga_end          ;No --> End routine

            ;-- If function is supported, BH contains the active video -
            ;-- system code; BH contains the inactive video sys. code -

            mov     cx,bx              ;Move result to CX
            xor     bh,bh              ;Set BH to 0
            or      ch,ch              ;Just one video system?
            je      tvga_1            ;Yes --> Convey first system's code

            ;-- Convert code of second system -----

            mov     bl,ch              ;Move second system code to BL
            add     bl,bl              ;Add offset to table
            mov     ax,offset DGROUP:vios_tab[bx] ;Get code from table and
            mov     [di+2],ax          ;place in caller's structure
            mov     bl,cl              ;Move first system's codes to BL

            ;-- Convert code of first system -----

tvga_1:      add     bl,bl              ;Add offset to table
            mov     ax,offset DGROUP:vios_tab[bx] ;Get code from table and
            mov     [di],ax            ;place in caller's structure

            mov     frame.cga_possi,0 ;CGA test failed
            mov     frame.ega_possi,0 ;EGA test failed
            mov     frame.mono_possi,0 ;MONO still needs testing

            mov     bx,di              ;Address of active structure
            cmp     byte ptr [bx],MDA  ;Monochrome system available?
            je      do_tmono           ;Yes --> Execute MDA/HGC test

            add     bx,2               ;Address of inactive structure
            cmp     byte ptr [bx],MDA  ;Monochrome system available?
            jne     tvga_end           ;No --> End routine

do_tmono:    mov     word ptr [bx],0    ;Pretend that this system
            ;is still unavailable
            mov     frame.mono_possi,1 ;Execute monochrome test

tvga_end:    ret                      ;Back to caller

test_vga    endp

;-----
;-- TEST_EGA: Determines whether an EGA card is installed

test_ega     proc near

            mov     ah,12h             ;Function 12H
            mov     bl,10h             ;Sub-function 10H
            int     10h                ;Call EGA-BIOS
            cmp     bl,10h             ;Is the function supported?
            je      tega_end           ;No --> End routine

            ;-- When this function is supported, CL contains the EGA ---
            ;-- card's DIP switch settings                               ---

            mov     al,cl              ;Move DIP switch settings to AL
            shr     al,1               ;Shift one position to the right

```

```

        mov     bx,offset DGROUP:ega_dips ;Offset address of table
        xlat                     ;Move element AL from table to AL
        mov     ah,al             ;Move monitor type to AH
        mov     al,EGA            ;It's an EGA card
        call    found_it          ;Move data to vector

        cmp     ah,MONO           ;Connected to monochrome monitor?
        je      is_mono           ;Yes --> not MDA or HGC

        mov     frame.cga_possi,0 ;Cannot be a CGA card
        jmp     short tega_end     ;End routine

is_mono:  mov     frame.mono_possi,0 ;If EGA card is connected to a mono
                                         ;monitor, it can be installed as
                                         ;either an HGC or MDA

tega_end: ret                     ;Back to caller

test_ega  endp

;-----
;-- TEST_CGA: Determines whether a CGA card is installed

test_cga  proc near

        mov     dx,3D4h           ;CGA tests port addr. of CRTC addr.
        call    test_6845         ;reg., to see if 6845 is installed
        jc      tega_end         ;No --> End test

        mov     al,CGA            ;Yes --> CGA is installed
        mov     ah,COLOR          ;CGA has color monitor attached
        jmp     found_it          ;Transfer data to vector

test_cga  endp

;-----
;-- TEST_MONO: Checks for the existence of an MDA or HGC card

test_mono  proc near

        mov     dx,3B4h           ;Check port address of CRTC addr. reg.
        call    test_6845         ;with MONO to see if there's a 6845
                                         ;installed
        jc      tega_end         ;No --> End test

        ;-- If there is a monochrome video card installed, the ----
        ;-- following determines whether it's an MDA or an HGC ----

        mov     dl,0BAh           ;Read MONO status port using 3BAH
        in      al,dx             ;
        and     al,80h            ;Check bit 7 only and
        mov     ah,al            ;move to AH

        ;-- If contents of bit 7 change during one of the following
        ;-- readings, the card is handled as an HGC

test_hgc:  mov     cx,8000h        ;Maximum of 32768 loop executions
        in      al,dx            ;Read status port
        and     al,80h            ;Check bit 7 only
        cmp     al,ah            ;Contents changed?
        jne     is_hgc           ;Bit 7 = 1 --> HGC
        loop    test_hgc         ;Continue loop

        mov     al,MDA           ;Bit 7 <> 1 --> MDA
        jmp     set_mono         ;Set parameters

is_hgc:   mov     al,HGC          ;Bit 7 = 1 --> is HGC
set_mono:  mov     ah,MONO        ;MDA/HGC on mono monitor
        jmp     found_it         ;Set parameters

test_mono  endp

;-----
;-- TEST_6845: Sets carry flag if no 6845 exists in port address of DX

test_6845  proc near

```

```

        mov  al,0Ah          ;Register 10
        out  dx,al          ;Register number of CRTC address reg.
        inc  dx              ;DX now in CRTC data register

        in   al,dx          ;Get contents of register 10
        mov  ah,al          ;and move to AH

        mov  al,4Fh         ;Any value
        out  dx,al          ;Write to register 10

waitforit: mov  cx,100       ;Short delay loop--gives 6845 time
            loop waitforit   ;to react

        in   al,dx          ;Read contents of register 10
        xchg al,ah          ;Exchange AH and AL
        out  dx,al          ;Send old value

        cmp  ah,4Fh         ;Written value read?
        je   t6845_end      ;Yes --> End test

        stc                 ;No --> Set carry flag

t6845_end: ret              ;Back from caller

test_6845  endp

;-----
;-- FOUND_IT: Transfers video card type to AL and monitor type to  ----
;--                AH in the video vector                                ----

found_it  proc near

        mov  bx,di          ;Address of active structure
        cmp  word ptr [bx],0 ;Video system already onboard?
        je   set_data       ;No --> Data in active structure

        add  bx,2           ;Yes --> Inactive structure address

set_data: mov  [bx],ax       ;Place data in structure
        ret                ;Back to caller

found_it  endp

;-----

_text    ends              ;End of code segment
        end                ;End of program

```