

```

;*****;
;*                               V I O S P A                               *;
;*-----*
;* Task           : Creates a function for determining the type          *;
;*                 of video card installed on a system. This             *;
;*                 routine must be assembled into an OBJ file,           *;
;*                 then linked to a Turbo Pascal program.                *;
;*-----*
;* Author        : Michael Tischer                                       *;
;* Developed on   : 10/02/88                                              *;
;* Last update    : 02/18/92                                              *;
;*-----*
;* Assembly      : MASM VIOSPA;                                           *;
;*               ... Link to a Turbo Pascal program                     *;
;*               using the {$L VIOSPA} compiler directive               *;
;*****;

```

```

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

```

```

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

;Monitor constants
NO_MON    = 0      ;No monitor/unrecognized code
MONO      = 1      ;Monochrome monitor
COLOR     = 2      ;Color Monitor
EGA_HIRES = 3      ;High-resolution/multisync monitor

```

```

ANLG_MONO   = 4                               ;Monochrome analog monitor
ANLG_COLOR  = 5                               ;Analog color monitor

;== Data segment =====

DATA    segment word public                    ;Turbo data segment

DATA    ends

;== Code segment =====

CODE     segment byte public                  ;Turbo code segment

        assume cs:CODE, ds:DATA

public   getvios

;-- Initialized global variables must be placed in the code segment ---

vios_tab equ this word

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

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

```

```

        db VGA      , ANLG_COLOR ;VGA card/analog color monitor

ega_dips equ this byte

        ;-- Conversion table for EGA card DIP switches -----

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

;-----
;-- GETVIOS: Determines type(s) of installed video card(s) -----
;-- Pascal call : GetVios ( vp : ViosPtr ); external;
;-- Declaration : Type Vios = record VCard, Monitor: byte;
;-- Return Value: None

getvios proc near

sframe      struc                ;Stack access structure
cga_possi   db ?                 ;Local variables
ega_possi   db ?                 ;Local variables
mono_possi  db ?                 ;Local variables
bptr        dw ?                 ;BPTR
ret_adr     dw ?                 ;Return address of calling program
vp          dd ?                 ;Pointer to first VIOS structure
sframe      ends                 ;End of structure

frame       equ [ bp - cga_possi ] ;Address elements of structure

        push bp                  ;Push BP onto stack
        sub  sp,3                 ;Allocate memory for local variables
        mov  bp,sp               ;Move SP to BP

```

```

mov  frame.cga_poss1,1 ;Is it a CGA?
mov  frame.ega_poss1,1 ;Is it an EGA?
mov  frame.mono_poss1,1;Is it an MDA or HGC?

mov  di,word ptr frame.vp      ;Get offset addr. of structure
mov  word ptr [di],NO_VIOS     ;No video system or unknown
mov  word ptr [di+2],NO_VIOS   ;system found

call test_vga                 ;Test for VGA card
cmp  frame.ega_poss1,0 ;Or is it an EGA card?
je   gv1                      ;No --> Go to CGA test

gv1: call test_ega             ;Test for EGA card
cmp  frame.cga_poss1,0 ;Or is it a CGA card?
je   gv2                      ;No --> Go to MDA/HGC test

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

call test_mono                ;Test for MDA/HGC card

;-- Determine video configuration -----

gv3: cmp  byte ptr [di],VGA ;VGA card?
je   gvi_end                  ;Yes --> Active card indicated
cmp  byte ptr [di+2],VGA;VGA card part of secondary system?
je   gvi_end                  ;Yes --> Active card indicated

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

```

```

and    al,7                ;Only modes 0-7 are of interest
cmp    al,7                ;Mono card active?
jne    gv4                 ;No --> CGA or EGA mode

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

cmp    byte ptr [di+1],MONO ;Mono monitor in first structure?
je     gvi_end             ;Yes --> Sequence O.K.
jmp    short switch        ;No --> Switch 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]     ;Switch with second structure
        mov  [di],ax

gvi_end: add    sp,3         ;Add local variables from stack
        pop  bp            ;Pop BP off of stack
        ret  4             ;Clear variables off of stack;
                           ;Return to Turbo

getvios    endp

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

test_vga   proc near

        mov  ax,1a00h       ;Function 1AH, sub-function 00H
        int  10h            ;Call VGA-BIOS

```

```

cmp     al,lah                ;Function supported?
jne     tvga_end              ;No --> End routine

;-- If function is supported, BL contains the code of the --
;-- active video system, while BH contains the code of    --
;-- the inactive video system                               --

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

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

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

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

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

mov     frame.cga_possi,0    ;CGA test fail?
mov     frame.ega_possi,0    ;EGA test fail?
mov     frame.mono_possi,0   ;Test for mono

mov     bx,di                ;Address of active structure
cmp     byte ptr [bx],MDA     ;Monochrome system online?

```

```

        je    do_tmono          ;Yes --> Execute MDA/HGC test

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

do_tmono: mov    word ptr [bx],0 ;Emulate if this system
        ;isn't available
        mov    frame.mono_possi,1;Execute monochrome test

tvga_end: ret                  ;Return to caller

test_vga    endp

;-----
;-- TEST_EGA: Determine 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 this function supported?
        je     tega_end        ;No --> End routine

        ;-- If the function IS supported, CL contains the
        ;-- EGA card DIP switch settings

        mov    bl,cl           ;Move DIP switches to BL
        shr    bl,1            ;Shift one position to the right
        xor    bh,bh           ;Index high byte to 0
        mov     ah,ega_dips[bx] ;Get element from table

```

```

        mov     al,EGA                ;Is it an EGA card?
        call    found_it              ;Transfer data to the vector

        cmp     ah,MONO               ;Mono monitor connected?
        je      is_mono               ;Yes --> Not MDA or HGC

        mov     frame.cga_possi,0     ;No CGA card possible
        jmp     short tega_end         ;End routine

is_mono:  mov     frame.mono_possi,0   ;EGA can either emulate MDA or HGC,
                                         ;if mono monitor is attached

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               ;Port addr. of CGA's CRTC addr. reg.
        call    test_6845             ;Test for installed 6845 CRTC
        jc      tega_end               ;No --> End test

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

test_cga  endp

;-----

```



```

;-- TEST_MONO: Checks for MDA or HGC card

test_mono  proc near

    mov     dx,3B4h           ;Port addr. of MONO's CRTC addr. reg.
    call    test_6845         ;Test for installed 6845 CRTC
    jc      tega_end          ;No --> End test

    ;-- Monochrome video card installed -----
    ;--
    mov     dl,0BAh           ;MONO status port at 3BAh
    in      al,dx             ;Read status port
    and     al,80h            ;Separate bit 7 and
    mov     ah,al             ;move to AH

    ;-- If the contents of bit 7 in the status port change ---
    ;-- during the following readings, it is handled as an ---
    ;-- HGC ---

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

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

is_hgc:     mov     al,HGC      ;Bit 7 = 1 --> HGC
set_mono:   mov     ah,MONO     ;MDA and HGC set as mono screen
            jmp     found_it    ;Set parameters

```

```
test_mono endp
```

```
;-----  
;-- TEST_6845: Returns set carry flag if 6845 doesn't lie in the  
;-- port address in DX
```

```
test_6845 proc near
```

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

```
    in al,dx            ;Get contents of register 0AH  
    mov ah,al           ;and move to AH
```

```
    mov al,4Fh          ;Any value  
    out dx,al           ;Write to register 0AH
```

```
waitforit: loop waitforit ;Short wait loop to which  
;6845 can react
```

```
    in al,dx            ;Read contents of register 0AH  
    xchg al,ah          ;Exchange Ah and AL  
    out dx,al           ;Send value
```

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

```
    stc                ;No --> Set carry flag
```

```
t6845_end: ret          ;Back to caller
```

test_6845 endp

;-- FOUND_IT: Transfers type of video card to AL and type of ----
;-- monitor in 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

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