

```

*****
;*          V 8 0 6 0 P A . A S M          *;
;*-----*
;* Task      : Contains routines for operating in 800x600 *;
;*            graphics mode on a Super VGA card with 16 *;
;*            colors. *;
;*-----*
;* Author    : MICHAEL TISCHER *;
;* Developed on : 01/14/91 *;
;* Last update  : 01/14/91 *;
;*-----*
;* Assembly   : MASM /mx V8060PA; or TASM -mx V8060PA *;
;*            ... Link to V8060P.PAS *;
;*****
;

;== Constants ==
GC_INDEX      = 3ceh          ;Index register for graphics ctrl.
GC_READ_MAP   = 4             ;Number of read map register
GC_BIT_MASK   = 8             ;Number of bit mask register
GC_GRAPH_MODE = 5             ;Number of graphics mode register

;== Data segment ==
DATA segment word public      ;Must be initialized during runtime
DATA ends

;== Program ==
CODE segment byte public      ;Program segment
      assume cs:code, ds:data

;-- Public declarations -----
public  init800600            ;Initialize 800x600 mode
public  setpix                ;Set pixel
public  getpix                ;Get pixel color

;-- Data in code segment -----
      ;-- Code number of 800x600 mode for
      ;-- different Super VGA cards
modeno    db 6Ah, 58h, 29h, 54h, 16h, 79h
modenoend equ this byte

;-----
;-- INIT800600: Initializes 800x600 Super VGA graphics mode
;--           with 16 colors
;-- Call from TP: function init800600 : boolean;
;-- Return value: TRUE = Mode was initialized, FALSE = Error

init800600 proc near
      ;-- Try all modes from MODENO table, until
      ;-- BIOS accepts a mode

it1:    mov  si,offset modeno    ;Begin with first mode from table
      xor  ah,ah                ;Function 00H: Initialize mode
      mov  al,cs:[si]           ;Load code number from table
      int  10h                  ;Initialize mode
      mov  ah,0fh               ;Function 0FH: Read mode
      int  10h
      cmp  al,cs:[si]           ;Mode set?
      je   it2                  ;Yes --> O.K.

      ;-- Wrong code number, choose next one from table -----
      inc  si                    ;SI to next code number
      cmp  si,offset modenoend   ;Execute entire table?
      jne  it1                  ;No --> Play it again, Sam

      mov  al,0                 ;Yes --> End function with error
      ret                      ;Return to caller

it2:    ;-- Mode was initialized -----

      mov  al,1                 ;All O.K.
      ret                      ;Return to caller

init800600 endp                ;End of procedure

;-----
;-- SETPIX: Changes a pixel to a specific color
;-- Call from TP: setpix( x , y : integer; pcolor : byte );

```

```

setpix      proc near

sframe      struc                ;Structure for stack access
bp0          dw ?                ;Gets BP
ret_adr0     dw ?                ;Return address to caller
pcolor       dw ?                ;Color
y0           dw ?                ;Y-coordinate
x0           dw ?                ;X-coordinate
sframe      ends                ;End of structure

frame       equ [ bp - bp0 ]     ;Address structure elements

push        bp                  ;Prepare for parameter addressing
mov         bp,sp               ;through BP register

;-- First compute offset in video RAM and shift value -----

mov         ax,frame.y0         ;Load Y-coordinate
mov         dx,800 / 8          ;Multiply by line width
mul         dx
mov         bx,frame.x0         ;Load X-coordinate
mov         cl,bl               ;Store low byte for shift computation

shr         bx,1                ;Divide X-coordinate by eight
shr         bx,1
shr         bx,1
add         bx,ax               ;Add offset from multiplication to it

and         cl,7                ;Compute bit mask from X-coordinate
xor         cl,7
mov         ah,1
shl         ah,cl

mov         dx,GC_INDEX         ;Access to graphics controller
mov         al,GC_BIT_MASK      ;Write bit mask in bit mask
out         dx,ax               ;register

mov         ax,(02h shl 8) + GC_GRAPH_MODE ;Set write mode 2 &
out         dx,ax               ;read mode 0

mov         ax,0A000h           ;Segment address of video RAM
mov         es,ax               ;to ES

mov         al,es:[bx]          ;Load latch register
mov         al,byte ptr frame.pcolor ;Load pixel color
mov         es:[bx],al          ;write back to latch reg.

;-- Set default values in different registers of graphics
;-- controller that have been changed

mov         ax,(0FFh shl 8) + GC_BIT_MASK
out         dx,ax

mov         ax,(00h shl 8) + GC_GRAPH_MODE
out         dx,ax

pop         bp
ret         6                   ;Return to caller, remove
                                ;arguments from stack

setpix      endp                ;End of procedure

```

```

;-----
;-- GETPIX: Returns a pixel color
;-- Call from TP: x := getpix( x , y : integer );

```

```

getpix      proc near

sframe1     struc                ;Structure for stack access
bp1          dw ?                ;Gets BP
ret_adr1     dw ?                ;Return address to caller
y1           dw ?                ;Y-coordinate
x1           dw ?                ;X-coordinate
sframe1     ends                ;End of structure

frame       equ [ bp - bp1 ]     ;Address structure elements

push        bp                  ;Prepare for parameter addressing
mov         bp,sp               ;through BP register

;-- First compute offset in video RAM and shift value -----

mov         ax,frame.y1         ;Load Y-coordinate
mov         dx,800 / 8          ;Multiply by line width
mul         dx

```

```

mov     si,frame.x1      ;Load X-coordinate
mov     cx,si            ;Store for shift computation

shr     si,1             ;Divide X-coordinate by eight
shr     si,1
shr     si,1
add     si,ax            ;Add offset from multiplication to it

and     cl,7             ;Compute bit mask from X-coordinate
xor     cl,7
mov     ch,1
shl     ch,cl

mov     ax,0A000h        ;Segment address of video RAM
mov     es,ax            ;to ES

mov     dx,GC_INDEX      ;Access to graphics controller
mov     ax,(3 shl 8)+ GC_READ_MAP ;First read out
xor     bl,bl            ;plane #3

gpl:    out     dx,ax      ;Load read map register
mov     bh,es:[si]        ;Load value from latch register
and     bh,ch            ;Leave only desired pixels
neg     bh               ;Set bit 7 according to pixel
rol     bx,1             ;Bit 7 from BH to Bit 1 in BL reg.

dec     ah               ;Process next bitplane
jge     gpl              ;> or = Null? ---> Continue

mov     al,bl            ;Function result to AL

pop     bp
ret     4                ;Return to caller, remove
                        ;arguments from stack

getpix   endp            ;End of procedure

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

```