

```

;*****;
;*              V 8 0 6 0 C A . A S M              *;
;*-----*
;*   Task           : Contains various 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     : 03/03/92                                     *;
;*-----*
;*   Memory model    : SMALL                                       *;
;*-----*
;*   Assembly        : MASM /mx V8060CA      or      TASM -mx V8060CA *;
;*****;

```

```

IGROUP group _text           ;Program segment
DGROUP group const,_bss, _data ;Data segment
      assume CS:IGROUP, DS:DGROUP, ES:DGROUP, SS:DGROUP

CONST segment word public 'CONST';This segment handles all
CONST ends                      ;readable constants

_BSS segment word public 'BSS' ;This segment handles all uninitial-
_BSS ends                      ;ized static variables

_DATA segment word public 'DATA' ;This segment handles all initialized
                                ;global and static variables
_DATA ends

;== Constants =====

```

```

GC_INDEX      = 3ceh          ;Index register of 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 =====

_DATA segment word public 'DATA'

    ;-- Code number for 800x600 mode for different
    ;-- Super VGA cards

modeno      db 54h, 6Ah, 58h, 29h, 16h, 79h
modenoend   equ this byte

_DATA ends

;== Program =====

_TEXT segment byte public 'CODE' ;Program segment

;-- Public declarations -----

public      _init800600        ;Initialize 800x600 mode
public      _setpix            ;Set pixel
public      _getpix            ;Get pixel color
public      _getfontptr        ;Return pointer to 8x8 font

;-----
;-- INIT800600: Initializes 800x600 Super VGA graphics mode
;--           with 16 colors
;-- Declaration : int init800600( void );

```

;-- Return value: 1 = Mode was initialized, 0 = Error

_init800600 proc near

;-- Try all modes from the modeno table, until BIOS
;-- accepts a mode

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

;-- Wrong code number, select next from table -----

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

xor ax,ax ;Yes --> End function with error
ret ;Return to caller

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

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

_init800600 endp ;End of procedure

```

;-- SETPIX: Changes a pixel to a specific color -----
;-- Declaration : void setpix( int x, int y, unsigned char pcolor );

_setpix    proc near

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

_setpix     endp                                ;End of procedure

```

```

;-- GETPIX: Returns a pixel color -----
;-- Declaration : unsigned char getpix( int x, int y );

_getpix    proc near

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

frame      equ [ bp - bpl ]                    ;Address structure elements

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

push si

;-- 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
        xor    bl,bl              ;plane #3

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

        dec     ah                ;Process next bit plane
        jge     gpl              ;> or = 0? ---> Continue

        mov     al,bl             ;Function result to AL

        pop     si
        pop     bp
        ret                      ;Return to caller

_getpix    endp                  ;End of procedure

;-- GETFONTPTR: Returns FAR pointer to 8x8 font table -----
;-- Declaration : void far * getfontptr( void )

```

```

_getfontptr proc near

    push    bp                ;Push BP onto stack

    mov     ax,1130h          ;Get register for function call
    mov     bh,3
    int     10h               ;Call BIOS video interrupt

    mov     dx,es              ;Pointer ES:BP returned in DX:AX
    mov     ax,bp

    pop     bp                ;Pop BP from stack
    ret                     ;Return to caller

_getfontptr endp              ;End of procedure

;== End =====

_text    ends                ;End program segment
end      ;End program

```