

```

;*****
;*                               V 1 6 C O L P A . A S M                               *;
;*-----*
;* Task                : Contains various routines for operating in          *;
;*                     : EGA and VGA graphics mode in 16 colors              *;
;*-----*
;* Author              : MICHAEL TISCHER                                     *;
;* Developed on        : 12/05/90                                           *;
;* Last update        : 01/14/91                                           *;
;*-----*
;* Assembly            : MASM V16COLPA.ASM; or TASM V16COLPA.ASM            *;
;*****

```

```

;== Constants =====

```

```

SC_INDEX      = 3c4h                ;Index register for sequencer ctrl.
SC_MAP_MASK   = 2                   ;Number of map mask register
SC_MEM_MODE   = 4                   ;Number of memory mode register

GC_INDEX      = 3ceh                ;Index register for graphics ctrl.
GC_FN_SELECT  = 3                   ;Number of function select reg.
GC_READ_MAP   = 4                   ;Number of read map register
GC_GRAPH_MODE = 5                   ;Number of graphics mode register
GC_MISCELL    = 6                   ;Number of miscellaneous register
GC_BIT_MASK   = 8                   ;Number of bit mask register

CRTC_INDEX    = 3d4h                ;Index register for CRT controller
CC_MAX_SCAN   = 9                   ;Number of maximum scan line reg.
CC_START_HI   = 0Ch                 ;Number of high start register
CC_UNDERLINE  = 14h                 ;Number of underline register
CC_MODE_CTRL  = 17h                 ;Number of mode control register

```

```

DAC_WRITE_ADR  = 3C8h           ;DAC write address
DAC_READ_ADR   = 3C7h           ;DAC read address
DAC_DATA       = 3C9h           ;DAC data register

VERT_RESCAN    = 3DAh           ;Input status register #1

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

vio_seg  dw (?)                  ;Video segment with current page
lnwidth  dw (?)                  ;Width of a pixel line in bytes
pageofs  dw (?)                  ;Page offset in the segment address

DATA     ends

;== Program =====
CODE      segment byte public    ;Program segment

        assume cs:code, ds:data

;-- Public declarations -----
public   init640480              ;Initialize 640x480 mode
public   init640350              ;Initialize 640x350 mode
public   init640200              ;Initialize 640x200 mode
public   init320200              ;Initialize 320x200 mode
public   setpix                  ;Set pixel
public   getpix                  ;Get pixel color
public   showpage                ;Display page 0 or 1
public   setpage                 ;Set page for setpix and getpix

```

```
;-----  
;-- INIT640350: Initializes 640x350 EGA graphics mode for 16 colors  
;-- Call from TP:  init640350;
```

```
init640350 proc near
```

```
        mov     al,10h           ;Set mode 10H  
        mov     cx,28000 / 16    ;Page offset  
  
init16:  mov     bx,640/8         ;Line width  
  
init:    mov     lnwidth,bx       ;Save line width  
        mov     pageofs,cx       ;Store page offset for segment addr.  
  
        xor     ah,ah           ;Call function 00H for setting  
        int     10h            ;mode  
  
        mov     vio_seg,0A000h   ;Set segment address of video RAM  
        ret                   ;Return to caller  
  
init640350 endp                 ;End of procedure
```

```
;-----  
;-- INIT640480: Initializes 640x480 VGA graphics mode with 16 colors  
;-- Call from TP:  init640480;
```

```
init640480 proc near
```

```
        mov     al,12h           ;Set mode 12H  
  
        ;-- Page offset unimportant, since only one page
```

```

        ;-- can be displayed

        jmp     init16

init640480 endp                                ;End of procedure

;-----
;-- INIT640200: Initializes 640x200 EGA graphics mode with 16 colors
;-- Call from TP:  init640200;

init640200 proc near

        mov     al,0Eh                ;Set 0EH mode
        mov     bx,640/8              ;Line width
        mov     cx,( 64000 / 4 ) / 16 ;Page offset
        jmp     init16

init640200 endp                                ;End of procedure

;-----
;-- INIT320200: Initializes 320*200 pixel mode
;-- Call from TP:  init320200;

init320200 proc near

        mov     al,0Dh                ;Set 0DH mode
        mov     bx,320/8              ;Line width
        mov     cx,( 32000 / 4 ) / 16 ;Page offset
        jmp     init

init320200 endp                                ;End of procedure

```

```

;-----
;-- SETPIX: Changes a pixel to a certain 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 ]      ;Addresses structure elements

push        bp                    ;Prepare for 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,linewidth          ;Multiply by line width
mul         dx
mov         bx,frame.x0           ;Load X-coordinate
mov         cl,bl                 ;Store Lo byte for shift calculation

shr         bx,1                   ;Divide X-coordinates 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-coordinates
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,vio_seg         ;Set ES to video RAM
mov     es,ax              ;

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

;-- Set default values in various registers of the 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 ]      ;Addresses 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-coordinates
mov dx,lnwidth  ;multiply times line width
mul dx
mov si,frame.x1 ;Load X-coordinates
mov cx,si       ;Store shift calculation

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

```

```

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

        mov    ax,vio_seg         ;Set ES to video RAM
        mov    es,ax              ;

        mov    dx,GC_INDEX        ;Load graphics controller index
        mov    ax,(3 shl 8)+ GC_READ_MAP ;First read
        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              ;Rotate bit 7 from BH to bit 1 in BL

        dec     ah                ;Process next bitplane
        jge     gpl              ;>= 0? --> Continue

        mov     al,bl             ;Function result to AL

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

getpix   endp                    ;End of procedure

```

```

;-----
;-- SETPAGE: Sets page for access from setpix and getpix

```



```
;-- Call from TP: setpage( page : byte );
```

```
setpage    proc near
```

```
    pop     cx                ;Pop return address from stack
    pop     ax                ;Pop argument from stack

    push    cx                ;Push return address back
    mul     pageofs           ;Multiply page number by page offset

    add     ax,0A000h          ;Add base segment address to it.
    mov     vio_seg,ax         ;Store new segment address

    ret                     ;Return to caller, argument already
                           ;removed from stack
```

```
setpage    endp                ;End of procedure
```

```
;-----
;-- SHOWPAGE: Display one of the two screen pages
;-- Call from TP: showpage( page : byte );
```

```
showpage   proc near
```

```
    pop     cx                ;Pop return address from stack
    pop     ax                ;Pop argument from stack

    push    cx                ;Push only return address
    mul     pageofs           ;Multiply page number by page offset
    mov     cl,4              ;Multiply all of it by 16
    shl     ax,cl
```

```

        mov     bl,al                ;Store Lo byte

        mov     dx,CRTC_INDEX       ;Address CRT controller
        mov     al,CC_START_HI      ;Move register number
        out     dx,ax
        inc     al                   ;Now output Lo byte
        mov     ah,bl
        out     dx,ax

        ;-- Wait to return to starting screen design -----

sp3:     mov     dx,VERT_RESCAN      ;Wait for end of
        in      al,dx               ;vertical rescan
        test    al,8
        jne     sp3

sp4:     in      al,dx               ;Go to start of rescan
        test    al,8
        je      sp4

        ret                        ;Return to caller

showpage  endp                    ;End of procedure

;== End =====

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

```