

```
;*****
;*          V 1 6 C O L C A . A S M          *;
;*-----*
;* Task      : Contains various routines for operating in *;
;*           EGA and VGA graphics modes in 16 colors. *;
;*-----*
;* Author    : Michael Tischer *;
;* Developed on : 12/05/90 *;
;* Last update  : 02/27/92 *;
;*-----*
;* Memory model : SMALL *;
;*-----*
;* Assembly    : MASM /mx V16COLCA.ASM; or TASM -mx V16COLCA *;
;*****
```

```
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';Readable constants
CONST ends
```

```
_BSS segment word public 'BSS' ;Un-initialized static variables
_BSS ends
```

```
_DATA segment word public 'DATA' ;Initialized global and static vars.
```

```
_DATA ends
```

```
== 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
```

```
PIXX          = 640       ;Horizontal resolution
```

```
== Data ==
```

```
_DATA segment word public 'DATA'
```

```
vio_seg       dw 0A000h    ;Video segment with current page
lnwidth       dw 0         ;Width of a pixel line in bytes
pageofs       dw 0         ;Page offset in the segment address
```

```
_DATA ends
```

```
== Program ==
```

```
_TEXT segment byte public 'CODE' ;Program segment
```

```
-- Public declarations -----
```

```
public _init640350      ;Initialize 640x350 mode
public _init640480      ;Initialize 640x480 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 or getpix
public  _getfontptr      ;Get pointer to 8x8 font

;-----
;-- INIT640350: Initializes 640x350 EGA graphics mode with 16 colors.
;-- Declaration : void init640350( void );

_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

    ret                     ;Return to caller

_init640350 endp          ;End of procedure

;-----
;-- INIT640480: Initializes 640x480 VGA graphics mode with 16 colors.
;-- Declaration : void init640480( void );

_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.
;-- Declaration : void init640200( void );

_init640200 proc near

    mov     al,0Eh        ;Set mode 0EH
    mov     cx,( 64000 / 4 ) / 16 ;Page offset

    jmp     init16

_init640200 endp          ;End of procedure

;-----
;-- INIT320200: Initializes 320x200 EGA graphics mode with 16 colors.
;-- Declaration : void init320200( void );

_init320200 proc near

    mov     al,0Dh        ;Set mode 0DH
    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 -----
;-- 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 ]      ;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,lnwidth      ;Multiply by line width
mul     dx
mov     bx,frame.x0      ;Load X-coordinate
mov     cl,bl            ;Store low byte for shift calculation

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

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 register

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

frame   equ [ bp - bp1 ]      ;Addresses structure elements

push    bp      ;Prepare for 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,lnwidth      ;Multiple by line width
    mul     dx
    mov     si,frame.x1     ;Load X-coordinate
    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 multiplication

    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     ;Access to graphics controller
    mov     ax,(3 shl 8)+ GC_READ_MAP ;Read plane #3
    xor     bl,bl           ;first

gp1:      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     gp1            ;>= 0? --> Continue

    mov     al,bl          ;Function result to AL

    pop     si
    pop     bp
    ret                   ;Return to caller

_getpix    endp           ;End of procedure

;-- SETPAGE: Sets page for access from setpix and getpix -----
;-- Declaration : void setpage( int page );

_setpage   proc near

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

    push    ax              ;Push these onto stack
    push    cx
    mul     pageofs         ;Multiple page number and page offset

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

    ret                   ;Return to caller, remove
                        ;arguments from stack

_setpage   endp           ;End of procedure

;-- SHOWPAGE: Display one of the two screen pages -----
;-- Declaration : void showpage( int page );

_showpage  proc near

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

    push    ax              ;Push these onto stack
    push    cx

    mul     pageofs         ;Multiply page number and page offset
    mov     cl,4            ;Multiply all by 16
    shl     ax,cl

```

```

        mov     bl,al                ;Store low byte

        mov     dx,CRTC_INDEX       ;Address CRT controller
        mov     al,CC_START_HI      ;Display high byte
        out     dx,ax
        inc     al                  ;Display low byte
        mov     ah,bl
        out     dx,ax

        ;-- Wait for start of screen design -----

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

sp4:    in      al,dx                ;Wait for start of rescan
        test    al,8
        je      sp4

        ret                        ;Return to caller

_showpage endp                    ;End of procedure

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

_getfontptr proc near

        push    bp                  ;Store BP

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

        mov     dx,es               ;Return pointer ES:BP 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 of program segment
        end                        ;End program

```