

```

;*****;
;*          V 3 2 2 0 C A . A S M          *;
;*-----*
;*   Task          : Contains routines for operating in      *;
;*                 320x200 256 color mode on a VGA card.    *;
;*-----*
;*   Author        : Michael Tischer                        *;
;*   Developed on   : 09/05/90                               *;
;*   Last update    : 02/13/92                               *;
;*-----*
;*   Assembly      : MASM /mx V3220CA;      or      TASM -mx V3220CA *;
;*                 ... Link to V3220C.C      *;
;*****;

```

```

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 =====

```

```

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_READ_MAP      = 4                ;Number of read map register
GC_GRAPH_MODE    = 5                ;Number of graphics mode register
GC_MISCELL       = 6                ;Number of miscellaneous 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             = 320              ;Horizontal resolution

;== Data =====

_DATA segment word public 'DATA'

vio_seg dw 0a000h                  ;Video segment with current page

_DATA ends

;== Program =====

```



```

_TEXT segment byte public 'CODE' ;Program segment

;-- Public declarations -----

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
public    _getfontptr                ;Return pointer to 8x8 font

;-- INIT320200: Initializes 320x200 pixel mode -----
;-- Declaration : void init320200( void );

_init320200 proc near

    ;-- Sets mode 13H, since BIOS uses this mostly for
    ;-- initialization.

    mov     ax,0013h                ;Set normal mode 13H
    int     10h

    mov     dx,GC_INDEX              ;Memory division
    mov     al,GC_GRAPH_MODE         ;Disable bit 4 of
    out     dx,al                    ;graphics mode register in
    inc     dx                        ;graphics controller
    in      al,dx
    and     al,11101111b
    out     dx,al
    dec     dx

    mov     al,GC_MISCELL            ;And change bit 1

```



```

out    dx,al                ;in the miscellaneous
inc    dx                  ;register
in     al,dx
and    al,11111101b
out    dx,al

mov     dx,SC_INDEX        ;Modify memory mode register in
mov     al,SC_MEM_MODE     ;sequencer controller so no further
out     dx,al              ;address division follows in
inc     dx                  ;bitplanes, and set the bitplane
in      al,dx              ;currently in the
and     al,11110111b       ;bit mask register
or      al,4
out     dx,al

mov     ax,vio_seg         ;Fill all four bitplanes with color
mov     es,ax              ;code 00H and clear the screen
xor     di,di
mov     ax,di
mov     cx,8000h
rep     stosw

mov     dx,CRTC_INDEX      ;Set double word mode using bit 6
mov     al,CC_UNDERLINE    ;in underline register of
out     dx,al              ;CRT controller
inc     dx
in      al,dx
and     al,10111111b
out     dx,al
dec     dx

mov     al,CC_MODE_CTRL     ;Using bit 6 in mode control reg.

```



```

        out    dx,al                ;of CRT controller, change
        inc    dx                  ;from word mode to byte mode
        in     al,dx
        or     al,01000000b
        out    dx,al

        ret                        ;Return to caller

_init320200 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

        push   di                  ;Push DI onto stack

        mov    ax,PIXX / 4        ;Compute offset in video RAM
        mul    frame.y0           ;and pass to DI

```



```

mov     cx,frame.x0
mov     bx,cx
shr     bx,1
shr     bx,1
add     ax,bx
mov     di,ax

and     cl,3                ;Compute bit mask for map to be
mov     ah,1                ;addressed, move to AH
shl     ah,cl
mov     al,SC_MAP_MASK     ;Register number to AL
mov     dx,SC_INDEX        ;Load sequencer index address
out     dx,ax              ;Load bit mask register

mov     ax,vio_seg         ;Set ES to video RAM
mov     es,ax
mov     al,byte ptr frame.pcolor ;Load pixel color and
stosb                          ;write to selected bitmap

pop     di                  ;Get registers from stack
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
xl       dw ?           ;X-coordinate
yl       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           ;Push SI onto stack

        mov  ax,PIXX / 4  ;Compute offset in video RAM
        mul  frame.yl     ;and pass to SI
        mov  si,frame.xl
        mov  cx,si
        shr  si,1
        shr  si,1
        add  si,ax

        and  cl,3         ;Compute bit mask for map to be
        mov  ah,cl        ;addressed, move to AH
        mov  al,GC_READ_MAP ;Register number to AL
        mov  dx,GC_INDEX  ;Load graphics controller index
        out  dx,ax        ;Load read map register

        mov  ax,vio_seg   ;Set ES to video RAM
        mov  es,ax
        mov  al,es:[si]   ;Load pixel color

        pop  si           ;Get registers from stack

```



```

        pop    bp

        ret                ;Return to caller

_getpix    endp                ;End of procedure

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

_setpage    proc near

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

        push   cx                ;Push these back on
        push   bx

        mov     al,4              ;High byte seg. addr = page * 4 + A0H
        mul     cl
        or      al,0A0h

        mov     byte ptr vio_seg + 1,al ;Move new segment address

        ret                ;Return to caller

_setpage    endp                ;End of procedure

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

_showpage    proc near

```



```

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

        push   cx                ;Push these back on
        push   bx

        mov     al,64            ;High byte of offset = page * 64
        mul     cl
        mov     ah,al           ;Move high byte of offset to AH

        ;-- Load new starting address -----

        mov     dx,CRTC_INDEX    ;Address CRT controller
        mov     al,CC_START_HI    ;Move register number
        out     dx,ax            ;to AL and exit

        ;-- 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

;-- 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
```