

```

;*****;
;*              S 6 4 3 5 C A . A S M              *;
;*-----*
;*   Task           : Contains routines for generating sprites in   *;
;*                   640x350 mode on EGA or VGA cards.             *;
;*-----*
;*   Author          : Michael Tischer                             *;
;*   Developed on     : 09/08/90                                     *;
;*   Last update      : 02/26/92                                    *;
;*-----*
;*   Memory model    : SMALL                                        *;
;*-----*
;*   Assembly        : MASM /mx S6435CA;   or   TASM -mx S6435CA   *;
;*                   ... Link to S6435C.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' ;All readable constants
CONST ends

```

```

_BSS segment word public 'BSS' ;All uninitialized static vars.
_BSS ends

```

```

_DATA segment word public 'DATA' ;All initialized global & 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_BIT_MASK   = 8             ;Number of bit mask register
```

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

```
== Program =====
```

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

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

```
public      _copybuf2video
public      _mergeandcopybuf2video
public      _copyvideo2buf
```

```
-----
```

```
-- CopyBuf2Video: Copies a rectangular range from video RAM
```

```
-- Declaration: CopyBuf2Plane( byte *bufptr,
--                               byte topage,
--                               int  tox,
--                               int  toy,
--                               byte rwidth,
--                               byte rheight );
```

```
_copybuf2video proc near
```

```
sfr0          struc          ;Structure for stack access
```

```

bp0      dw ?           ;Gets BP
stofs0   dw ?           ;Local var.: Starting offset in video RAM
ret_adr0 dw ?           ;Return address to caller
bufptr0  dw ?           ;Buffer pointer
topage   dw ?           ;To page
tox      dw ?           ;To X-coordinate
toy      dw ?           ;To Y-coordinate
rwidth0  dw ?           ;Range width
rheight0 dw ?           ;Range height
sfr0     ends          ;End of structure

```

```

fr      equ [ bp - bp0 ] ;Addr. of structure elements
bfr     equ byte ptr [ bp - bp0 ];Addr. of stack elements as bytes
sub     sp,2             ;Space for local variables

```

```

push  bp      ;Prepare BP register for
mov   bp,sp   ;addressing parameters
push  si      ;Store SI and DI
push  di

```

```

;-- Compute segment address for video RAM access -----

```

```

mov  ah,0A0h      ;Move ES to start of T0 page
cmp  bfr.topage,0 ;Page 0?
je   cv0          ;Yes --> AL is O.K.

```

```

mov  ax,0A6D6h    ;No --> Page 1 from A6D6H

```

```

cv0:      mov  es,ax

```

```

;-- Compute offset for target position in page -----

```

```

mov     ax,PIXX / 8           ;Move AX to TO position
mul     fr.toy
mov     bx,fr.tox
shr     bx,1
shr     bx,1
shr     bx,1
add     bx,ax
mov     fr.stofs0,bx         ;Store starting offset in local vars.

mov     si,fr.bufptr0        ;Set DS:SI to buffer

;-- Load copy loop counter -----

mov     dl,bfr.rwidth0       ;DL = Bytes
mov     bx,PIXX / 8          ;BX as offset to next row
sub     bl,dl
xor     ch,ch                 ;Counter high byte is always 0

;-- Prepare bitplane to be addressed -----

mov     ah,1                 ;Load plane number as bit mask
mov     al,SC_MAP_MASK       ;Move register no. to AL

cvl:    mov     dx,SC_INDEX    ;End access to
out     dx,ax                ;plane

;-- Bitplane copy routine, ignore background -----

mov     di,fr.stofs0         ;Move DI to starting offset
mov     dh,bfr.rheight0      ;DH = Lines
mov     dl,bfr.rwidth0       ;DL = Bytes

```

```

cv2:      mov     cl,dl          ;Move number of bytes to CL

rep movsb          ;Copy line
add     di,bx       ;Add DI to next line
dec     dh          ;One line remaining?
jne     cv2         ;Yes --> Continue

shl     ah,1        ;Toggle to next plane
test    ah,16       ;All planes processed?
je      cv1         ;No --> Process next plane

mov     ax,(0Fh shl 8)+ SC_MAP_MASK ;Enable access to
mov     dx,SC_INDEX ;all bitplanes
out     dx,ax

pop     di          ;Pop DI and SI from stack
pop     si
pop     bp          ;Pop BP from stack

add     sp,2        ;Remove local variables
ret                     ;Return to caller

```

_copybuf2video endp

```

;-----
;-- MergeAndCopyBuf2Video: A bit mask links the contents of a back-
;--                        ground buffer to the contents of a sprite
;--                        buffer and copies the result to video RAM
;-- Declaration : MergeAndCopyBuf2Video( void * spribufptr,
;--                                     void * hgbufptr,
;--                                     void * andbufptr,
;--                                     BYTE page,

```

```

;--                                int  tox,
;--                                int  toy,
;--                                BYTE rwidth,
;--                                BYTE rheight );
;-- Info                          : see CopyVideo2Buf

```

_mergeandcopybuf2video proc near

```

sfr2      struc                ;Structure for stack access
bp2        dw ?                ;Gets BP
andptr2    dw ?                ;Local var.: Pointer in AND buffer
stofs2     dw ?                ;Local var.: Starting offset in video RAM
ret_adr2   dw ?                ;Return address to caller
spribufptr dw ?                ;Pointer to sprite buffer
hgbufptr   dw ?                ;Pointer to background buffer
andbufptr  dw ?                ;Pointer to AND buffer
topage2    dw ?                ;To page
tox2       dw ?                ;To X-coordinate
toy2       dw ?                ;To Y-coordinate
rwidth2    dw ?                ;Range width
rheight2   dw ?                ;Range height
sfr2       ends                ;End of structure

fr         equ [ bp - bp2 ]     ;Addresses structure elements
bfr        equ byte ptr [ bp - bp2 ] ;Addresses stack element as byte

sub    sp,4                    ;Space for local variables

push   bp                      ;Prepare BP register for
mov    bp,sp                   ;addressing parameters

push   ds                      ;Store DS

```

```

push  si                      ;Store SI and DI
push  di

;-- Calculate segment address for access to video RAM -----

mov    ax,0A000h              ;Move ES to start of TO page
cmp    bfr.topage2,0          ;Page 0?
je     cm0                    ;Yes --> AL already O.K.

mov    ax,0A6D6h              ;No --> page 1 of A6D6H

cm0:   mov    es,ax

;-- Computer offset for target position in page -----

mov    ax,PIXX / 8            ;Move AX to FROM target position
mul    fr.toy2
mov    bx,fr.tox2
shr    bx,1
shr    bx,1
shr    bx,1
add    bx,ax
mov    fr.stofs2,bx          ;Store starting offset in local vars.

;-- Load counter for copy loop -----

mov    dl,bfr.rwidth2        ;DL = Bytes
mov    bx,PIXX / 8           ;BX as offset to next line
sub    bl,dl
xor    ch,ch                  ;High byte of counter is always 0

;-- Configure bitplane to be addressed -----

```

```

        mov     ah,1             ;Load plane number as bit mask

cm1:     mov     al,SC_MAP_MASK   ;Move register number to AL
        mov     dx,SC_INDEX      ;Open access to plane for
        out     dx,ax            ;processing

        ;-- Copy routine for a bitplane without -----
        ;-- checking the background

        mov     dx,fr.andbufptr  ;Copy offset address of AND
        mov     fr.andptr2,dx    ;pointer to local variable
        mov     di,fr.stofs2     ;DI to start offset
        mov     dh,bfr.rheight2  ;DH = Lines
        mov     dl,bfr.rwidth2   ;DL = Bytes

cm2:     mov     cl,dl            ;Number of bytes to CL

cm3:     mov     si,fr.hgbufptr   ;Load pointer to background buffer
        lodsb                    ;Load byte from background buffer
        mov     fr.hgbufptr,si   ;Store incremented offset

        mov     si,fr.andptr2    ;Load pointer in AND buffer
        and     al,[si]          ;Link background and AND mask
        inc     si               ;Increment offset in AND buffer
        mov     fr.andptr2,si    ;and save

        mov     si,fr.sprbufptr  ;Load pointer to sprite buffer
        or      al,[si]          ;OR byte from sprite buffer
        inc     si               ;Increment offset in sprite buffer
        mov     fr.sprbufptr,si  ;and save

```



```

    stosb                ;Write byte to video RAM
loop   cm3               ;Process next byte

    add    di,bx         ;DI in next line
    dec    dh            ;Another line?
    jne    cm2           ;Yes --> Continue

    shl    ah,1          ;Switch to next plane
    test   ah,16         ;All planes processed?
    je     cm1           ;No --> Continue with next plane

    mov     ax,(0Fh shl 8)+ SC_MAP_MASK    ;Allow access to
    mov     dx,SC_INDEX                    ;all bitplanes
    out     dx,ax

    pop     di                ;Pop DI and SI from stack
    pop     si
    pop     ds                ;Pop DS from stack
    pop     bp

    add     sp,4              ;Remove local variables
    ret                     ;Return to caller

```

```

_mergeandcopybuf2video endp

```

```

;-----
;-- CopyVideo2Buf: Copies a rectangular range from video RAM to
;--                a buffer
;-- Declaration : CopyPlane2Buf( byte *bufptr,
;--                               byte frompage,
;--                               int  fromx,
;--                               int  fromy,

```

```

;--                                byte rwidth,
;--                                byte rheight );
;-- Info      : In this version of the routine, the area to be
;--            copied must begin at a pixel column that can be
;--            divided by eight and extend over a multiple of
;--            eight pixels.
;--            RWIDTH refers to the number of bytes per line
;--            on a bitplane

```

```

_copyvideo2buf proc near

```

```

sfrl      struc                ;Structure for stack access
bpl       dw ?                 ;Gets BP
stofsl    dw ?                 ;Local var.: Starting offset in video RAM
ret_adrl  dw ?                 ;Return address to caller
bufptrl   dw ?                 ;Buffer pointer
frompage  dw ?                 ;From page
fromx     dw ?                 ;From X-coordinate
fromy     dw ?                 ;From Y-coordinate
rwidthl   dw ?                 ;Range width of range in pixels
rheightl  dw ?                 ;Range height in pixel lines
sfrl      ends                 ;End of structure

fr        equ [ bp - bpl ]      ;Addresses structure elements
bfr       equ byte ptr [ bp - bpl ] ;Addresses stack element as byte

sub       sp,2                 ;Space for local variables

push     bp                     ;Prepare BP register for
mov      bp,sp                 ;addressing parameters

push     ds                     ;Store DS

```

```

push  si                ;Push SI and DI onto stack
push  di

push  ds                ;Prepare ES for buffer access
pop   es                ;with DS

;-- Calculate segment address for access to video RAM -----

mov   ax,0A000h         ;Move ES start of FROM page
cmp   bfr.frompage,0    ;Page 0?
je    cc0               ;Yes --> AL is O.K.

mov   ax,0A6D6h         ;No --> Page 1 of A6D6H

cc0:   mov   ds,ax

;-- Form offset in page to be read -----

mov   ax,PIXX / 8       ;Move AX to target position FROM
mul   fr.fromy
mov   bx,fr.fromx
shr   bx,1
shr   bx,1
shr   bx,1
add   bx,ax
mov   fr.stofs1,bx      ;Store start offset in local variable

mov   di,fr.bufptr1     ;Set ES;DI to buffer

;-- Load counter for copy loop -----

mov   dl,bfr.rwidth1    ;DL = Bytes

```

```

mov    bx,PIXX / 8          ;BX as offset to next line
sub    bl,dl
xor     ch,ch                ;High byte of counter is always 0

;-- Set addressable bitplane -----

xor     ah,ah                ;Begin with plane #0
mov     al,GC_READ_MAP       ;Register number to AL

ccl:    mov    dx,GC_INDEX     ;Load index address of graphic ctrl.
        out    dx,ax          ;Load read map register

;-- Copy routine for a bitplane without checking -----
;-- background

mov     dh,bfr.rheight1      ;DH = Lines
mov     dl,bfr.rwidth1       ;DL = Bytes
mov     si,fr.stofsl         ;Load starting offset to SI

cc2:    mov     cl,dl          ;Number of bytes to CL

        rep movsb             ;Copy line
        add     si,bx          ;SI in next line
        dec     dh             ;Another line?
        jne     cc2           ;Yes --> Continue

        inc     ah             ;Switch to next plane
        cmp     ah,4           ;All planes processed?
        jne     cc1           ;No --> Continue with next plane

        pop     di             ;Pop DI and SI from stack
        pop     si

```

```

        pop    ds                ;Pop DS from stack

        pop    bp

        add    sp,2              ;Remove local variables
        ret                    ;Return to caller

_copyvideo2buf endp

;== End =====

_text    ends                    ;End of program segment
        end                    ;End program

```