

```

;*****;
;*          S 3 2 4 0 P A . A S M          *;
;*-----*
;* Task          : Contains routines for generating sprites in *;
;*               : 320x400 256 color mode on a VGA card.      *;
;*-----*
;* Author        : Michael Tischer                          *;
;* Developed on   : 09/08/90                                  *;
;* Last update    : 02/26/92                                  *;
;*-----*
;* Assembly      : MASM /mx S3240PA;      or      TASM -mx S3240PA *;
;*               : ... link to S3240P.PAS                      *;
;*****;

```

```

;== 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

```

```

PIXX          = 320           ;Horizontal resolution

```

```

;== Data segment =====

```

```

DATA    segment word public

```

```

DATA    ends

```

```

;== Program =====

```

```

CODE      segment byte public      ;Program segment

      assume cs:code, ds:data

;-- Public declarations -----

public    copybuf2plane
public    copyplane2buf

;-----
;-- CopyBuf2Plane: Copies buffer contents to rectangular bitplane range
;-- Call from TP : CopyBuf2Plane( bufptr    : pointer;
;--                               topage    : byte;
;--                               tox,      : integer;
;--                               toy       : integer;
;--                               rwidth,   : byte;
;--                               rheight  : byte;
;--                               bg       : bool  );

copybuf2plane proc near

sfr0      struc                ;Structure for stack acces
bp0       dw ?                ;Gets BP
ret_adr0  dw ?                ;Return address from caller
bg        dw ?                ;Background
rheight0  dw ?                ;Range height
rwidth0   dw ?                ;Range width
toy       dw ?                ;To Y-coordinate
tox       dw ?                ;To X-coordinate
topage    dw ?                ;To page
bufptr0   dd ?                ;Buffer pointer

```

```

sfr0      ends                      ;End structure

fr        equ [ bp - bp0 ]          ;Addr. of structure elements
bfr       equ byte ptr [ bp - bp0 ] ;Addr. of stack elements as bytes

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

push ds
cld              ;Increment on string instructions

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

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

mov ah,0A8h      ;No --> Page 1 from A800H

cv0:        xor al,al              ;Low byte always null
mov es,ax

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

mov ax,PIXX / 4  ;Move DI to target of TO page
mul fr.toy
mov di,fr.tox
mov cx,di        ;Store X-coordinate for plane range
shr di,1
shr di,1
add di,ax

```

```

;-- Configure bitplane to be addressed -----
mov     ah,1                ;Configure plane number
and     cl,3                ;as a bit mask
shl     ah,cl
mov     dx,SC_INDEX         ;Store access to bitplane
mov     al,SC_MAP_MASK      ;to be processed
out     dx,ax

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

mov     dh,bfr.rheight0     ;DH = Rows
mov     dl,bfr.rwidth0      ;DL = Bytes
mov     bx,PIXX / 4         ;BX as offset to next row
sub     bl,dl
xor     ch,ch                ;Counter high byte is always 0

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

cmp     bfr.bg,0            ;Ignore background?
jne     cv2                  ;Yes --> Alternate copy routine

;-- Copy routine for bitplane, without background support --

cv1:     mov     cl,dl        ;Number of bytes to CL

rep movsb                ;Copy row
add     di,bx             ;Add DI to next row
dec     dh                ;One row remaining?
jne     cv1                ;Yes --> Continue

jmp     short cvend        ;No --> Copy entire buffer

```

```

        ;-- Copy routine for individual bitplanes using buffer -----
cv2:      mov     cl,dl                ;Number of bytes to CL

cv3:      lodsb                     ;Load byte from buffer
        cmp     al,255              ;Background byte?
        je      cv5                 ;Yes --> Exclude from copy
        stosb                      ;No --> Place in video RAM
        loop    cv3                 ;Process next page

cv4:      ;-- Switch video RAM pointer to next row -----
        add     di,bx               ;Add DI to next row
        dec     dh                  ;One row remaining?
        jne     cv2                 ;Yes --> Continue
        jmp     short cvend          ;No --> Copy entire buffer

cv5:      ;-- Background byte, do not include in copy -----
        inc     di                  ;This byte not described in video RAM
        loop    cv3                 ;Byte remaining in this row?
        jmp     cv4                 ;No --> Process next row

cvend:    pop     ds
        pop     bp

        ret     16                  ;Return to caller, remove
                                   ;parameters from stack
copybuf2plane endp

;-----

```

```

;-- CopyPlane2Buf: Copies rectangular bitplane range to a buffer
;-- Call from C : CopyPlane2Buf( bufptr   : pointer;
;--                               frompage  : byte;
;--                               fromx,    : integer;
;--                               fromy     : integer;
;--                               rwidth,   : integer;
;--                               rheight  : byte );

```

```

copyplane2buf proc near

```

```

sfrl      struc                ;Structure for stack acces
bpl       dw ?                 ;Gets BP
ret_adrl  dw ?                 ;Return address from caller
rheightl  dw ?                 ;Range height in pixel rows
rwidthl   dw ?                 ;Range width in pixels
fromy     dw ?                 ;From Y-coordinate
fromx     dw ?                 ;From X-coordinate
frompage  dw ?                 ;From page
bufptrl   dd ?                 ;Buffer pointer
sfrl      ends                 ;End structure

fr        equ [ bp - bpl ]      ;Addr. of structure elements
bfr       equ byte ptr [ bp - bpl ] ;Addr. of stack elements as bytes

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

        push ds

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

        mov  ah,0A0h           ;Move ES to start of FROM page

```

```

    cmp     bfr.frompage,0      ;Page 0?
    je      cc0                 ;Yes --> AL is O.K.

    mov     ah,0A8h             ;No --> Page 1 from A800H

cc0:   xor     al,al              ;Low byte always null
    mov     ds,ax

    ;-- Compute offset of page to be read -----

    mov     ax,PIXX / 4         ;Move SI to target of FROM
    mul     fr.fromy
    mov     si,fr.fromx
    mov     cx,si               ;Move coordinates to CX
    shr     si,1
    shr     si,1
    add     si,ax

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

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

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

    mov     dh,bfr.rheight1     ;DH = Rows
    mov     dl,bfr.rwidth1      ;DL = Bytes
    mov     bx,PIXX / 4         ;BX as offset to next row
    sub     bl,dl

```

```

        xor    ch,ch                ;Counter high byte is always 0

        les    di,fr.bufptr1      ;Set ES:DI to buffer

        ;-- Copy routine for bitplane, without background support --

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

        rep movsb                  ;Copy row
        add     si,bx              ;Add SI to next row
        dec     dh                 ;One row remaining?
        jne     ccl               ;Yes --> Continue

        pop     ds
        pop     bp

        ret     14

copyplane2buf endp

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

```