

```

;*****;
;*          S 3 2 4 0 C 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 S3240CA; or TASM -mx S3240CA *;
;*               : ... link to S3240C.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

PIXX             = 320                    ;Horizontal resolution

;== Program =====

_TEXT segment byte public 'CODE';Program segment

;-- Public declarations -----

public      _copybuf2plane
public      _copyplane2buf

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

_copybuf2plane proc near

sfr0            struc                      ;Structure for stack acces
bp0             dw ?                      ;Gets BP
ret_adr0        dw ?                      ;Return address from 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
bg        dw ?                ;Background
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 di
push si

;-- 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

mov    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,d1      ;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     si
        pop     di
        pop     bp

```

```

        ret                                ;Return to caller, remove
                                         ;parameters from stack
_copybuf2plane endp

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

_copyplane2buf proc near

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

pop     es
mov     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     si
pop     di
pop     bp

ret

_copyplane2buf endp

;== End =====

```



```
_text      ends      ;End code segment
end        ;End program
```