

```

;*****
;*                               X M S C A . A S M                               *;
;*-----*
;* Task      : Assembly routine for linking to the C program *;
;*            XMSC.C. Makes a routine for calling the XMS *;
;*            driver available. *;
;*            Implementation here for the SMALL memory *;
;*            model. *;
;*-----*
;* Author     : MICHAEL TISCHER *;
;* Developed on : 07/27/90 *;
;* Last update  : 07/27/90 *;
;*-----*
;* Assembly   : MASM XMSCA; or TASM XMSCA; *;
;*            ... then link with compiled C program XMSC.C *;
;*****

;== Segment declarations for the C program =====

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
extrn _XMSPtr : dword          ;Reference to the XMS pointer

_DATA ends

;== Program =====

_TEXT segment byte public 'CODE' ;Program segment

public _XMSCall

;-----
;-- XMSCall : General routine for calling an XMS function
;-- Call of C : void XMSCall( BYTE FktNr, XMSRegs *Xr ) with
;--            typedef struct { WORD AX, BX, DX, SI, Segment } XMSRegs;
;-- Return value: None
;-- Info : - Before calling this procedure, load only those registers
;--          necessary for calling the specified function.
;--          - After the XMS function call, the contents of the various
;--          processor registers are copied to the corresponding
;--          components of the passed structure.
;--          - Before this procedure is called for the first time, the
;--          XMSInit must have been called successfully.

_XMSCall proc near

sframe struc ;Structure for stack access
bp0 dw ? ;Gets BP
ret_adr dw ? ;Return address to caller
fktnr dw ? ;Number of XMS function
xrptra dw ? ;Pointer to register structure
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 si ;Push SI and DI onto stack
      push di

      mov cx,ds ;Store DS in CX
      push cx ;and push onto stack
      mov di,frame.xrptra ;Load function number
      mov ah,byte ptr frame.fktnr ;Load pointer to structure
      mov bx,[di+2] ;Load register from components

```

```

mov     dx,[di+4]                ;of structure
mov     si,[di+6]
mov     ds,[di+8]
mov     es,cx                    ;Load ES with DS
call    es:[_XMSPtr]             ;Call XMS handler
mov     cx,ds                    ;Store DS in CX
pop     ds                       ;Get old DS from stack
mov     di,frame.xrptr           ;Load pointer to structure
mov     [di],ax                  ;Enter registers in the
mov     [di+02],bx               ;components of the structure
mov     [di+04],dx
mov     [di+06],si
mov     [di+08],cx

pop     di                       ;Get SI and DI from stack
pop     si

pop     bp                       ;Get BP from stack
ret                                     ;Return to C program

```

```

_XMSCall    endp

```

```

;-----

```

```

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

```