

```

;*****;
;*                E X E S Y S . A S M                *;
;*-----*
;* Task           : This driver can be called either from DOS *;
;*                as an executable file, or from CONFIG.SYS *;
;*                as a device driver. *;
;*-----*
;* Author         : MICHAEL TISCHER *;
;* Developed on   : 11/01/1991 *;
;* Last update    : 01/07/1991 *;
;*-----*
;* Assembly       : MASM EXESYS; *;
;*                LINK EXESYS; *;
;*                or *;
;*                TASM EXESYS *;
;*                TLINK EXESYS *;
;*****;

```

```
code segment
```

```
assume cs:code,ds:code
```

```
org 0 ;The program has no PSP, so it
;begins at offset address 0
```

```
== Constants ==
```

```
cmd_fld equ 2 ;Command field offset in data block
status equ 3 ;Status field offset in data block
end_adr equ 14 ;Driver end address offset in data blk
```

```
== Data ==
```

```
-- Device driver header -----
```

```
dw -1,-1 ;Link to next driver
dw 1010000000000000b ;Driver attribute
dw offset strat ;Pointer to strategy routine
dw offset intr ;Pointer to interrupt routine
db "$EXESYS" ;Driver name
```

```
db_ptr dw (?), (?) ;Address of specified data block
```

```
=====
```

```
== EXE program routines ==
```

```
=====
```

```
exestart proc far
```

```
push cs ;Set DS to CS
pop ds

mov ah,09h ;Display message
mov dx,offset exemes
int 21h

mov ax,4C00h ;End program as usual
int 21h
```

```
exestart endp
```

```
exemes db "EXESYS - (c) 1991 by Michael Tischer", 13,10,10
db "Called as an EXE program!", 13, 10, "$"
```

```
=====
```

```
== Driver routines and functions ==
```

```
=====
```

```
strat proc far ;Strategy routine
```

```
mov cs:db_ptr,bx ;Set data block address in
mov cs:db_ptr+2,es ;the DB_PTR variable

ret ;Return to caller
```

```

strat      endp

;-----

intr       proc far                ;Interrupt routine

    push ax                        ;Place registers on stack
    push bx
    push cx
    push dx
    push di
    push si
    push bp
    push ds
    push es
    pushf                          ;Set flag register

    push cs                        ;Set data segment register (code
    pop  ds                        ;and data are identical)

    les di,dword ptr db_ptr;Data block address after ES:DI
    mov ax,8003h                   ;executed by error
    cmp byte ptr es:[di+cmd_fld],00h ;Only INIT is permitted
    jne short intr_end             ;Error --> Return to caller

    call init                      ;Can only be function 00H

    ;-- End execution of the function -----

intr_end label near

    or  ax,0100h                   ;Set ready bit
    mov es:[di+status],ax          ;Save entire contents in status field

    popf                           ;Pop flag register
    pop es                         ;Pop remaining registers
    pop ds
    pop bp
    pop si
    pop di
    pop dx
    pop cx
    pop bx
    pop ax

    ret                            ;Return to caller

intr      endp

;-----
;-- If called as a device driver,
;-- memory is then released

init      proc near                ;Initialization routine

    mov word ptr es:[di+end_adr],offset init ;Set end address
    mov es:[di+end_adr+2],cs              ;of driver

    mov ah,09h                          ;Display message
    mov dx,offset ddmes
    int 21h

    xor ax,ax                            ;Everything is O.K.
    ret                                  ;Return to caller

init      endp

;-- Data no longer needed after initialization -----

ddmes     db "EXESYS - (c) 1991 by Michael Tischer", 13,10,10
          db "Called as a device driver!", 13, 10, "$"

;=====

code      ends
end exestart

```