6. Praktikum

1. Aufgabe: DOS-Treiber für die Serielle Schnittstelle

Der TSR-Treiber INT 60h von Aufgabe 1 ist bei gleichen Randbedingungen in einen DOS-Treiber umzuprogrammieren und kann dann mit DOS-Funktionen angesprochen werden *(Skript DOS-Treiber)*
Dazu ist das TSR-Treiberprogramm an den vorgegebenen Formalismus für DOS-Treiber anzupassen.

Im **Anhang 1** ist der mit 5 markierten Lücken versehener **DOS-Treiber TREIBDOx.asm** für die serielle Schnittstelle mit **Empfangs-Interrupt** und eigenem **Empfangs-Ringpuffer** angefügt.
Die fünf durch Fragezeichen markierten Lücken sind für die Funktionsfähigkeit durch Programm-Sequenzen zu ergänzen.

Mit TREIBDOS.asm steht dann ein funktionsfähiger DOS-Treiber für die serielle Schnittstelle zur Verfügung, der einen interruptgetriebenen Empfang mit eigenem Ringpuffer aufweist.
Der DOS-Treiber wird fest auf die Schnittstelle COM1, kein Loopback und kein FIFO initialisiert und für andere Varianten bzw Übertragungsparameter ist eine Neuassemblierung erforderlich.
(die Möglichkeit einer Neu-Initialisierung des **residenten Treibers** könnte mit DOS-Multiplexer 2Fh realisiert werden)

TREIBDOS.exe ist mit EXE2BIN nach DRVSER.sys umzuwandeln und mit DEVICE = c:\...\DRVSER.sys in CONFIG.sys wird beim Booten der DOS-Treiber dann resident gemacht.

Hinweis: Ab WindowsMe steht kein realer DOS-Modus und damit kein CONFIG.sys mehr zur Verfügung.
Abhilfe unter WindowsMe mit DEVICE.COM siehe ***Skript DOS-Treiber Seite 2***
Unter WindowsXP noch keine Lösung des Problems bekannt. (eine Lösung gibt Bonus !!!!!!)

Beim Arbeiten in der DOS-Box vor dem Erstellen einer neuen DRVSER.sys-Version auf Windows-Ebene den alten DRVSER.sys löschen (geht in DOS-Box nicht)
Dann in der DOS-Box mit EXE2BIN den neuen DRVSER.sys erstellen, kurz nach Windows zurück und nach erneutem Aufruf der DOS-Box wird die neue Treiberversion installiert

2. Aufgabe: Terminalemulation mit DOS-Treiber Serielle Schnittstelle unter Benutzung der DOS-Datei-Handle-Funktionen und des DOS Einheitentreibers IOCTL

Die Terminalemulation vom 5.Praktikum Aufgabe 2 ist jetzt unter Verwendung des DOS-Treiber DRVSER.sys zu realisieren.
Der Zugriff auf den beim Booten resident gemachten DOS-Treiber für die serielle Schnittstelle ist durch die **DOS-Datei-Handle-Funktionen** 3Ch-43h INT 21h und den **DOS Einheitentreiber IOCTL** 44h INT 21h möglich.
Für das Testen des DOS-Treibers DRVSER.sys steht im Anhang 2 mit DRVTERM3.asm ein funktionsfähiges Terminalemulations-Programm als Beispiel zur Verfügung.

Skript DOS-Treiber Seite 10 -18

3. Aufgabe: Dateitransfer mit DOS-Treiber Serielle Schnittstelle

Die Datei-Übertragung vom 5.Praktikum 3.Aufgabe ist bei gleicher Aufgabenstellung jetzt unter Verwendung des **DOS-Treibers DRVSER.sys** zu realisieren.
Dazu sollte im Gegensatz zur Terminalemulation jetzt auch der Zugriff auf die zu übertragende Datei mit den **DOS-Datei-Handle-Funktionen** erfolgen.

Skript: Systemunterlagen, System-Funktionen

Anhang 1: TREIBDOX.asm

```
;TREIBDOx.asm DOS-Treiber- Senden und Empfangen  -serielle Schnittstelle
; basiert auf SERINT (INT 60h ) mit Empfangs-Interrupt und Ringpuffer
; TASM  TREIBDOx ; TLINK  TREIBDOx ;  EXE2BIN TREIBDOx.EXE DRVSER.SYS
; Treiber einbinden in CONFIG.SYS  mit DEVICE=c:\...DRVSER.SYS
; Parameter: COM1(fest) 9600 Bd, 8 Datenbit, gerade Paritaet, 2 Stoppbit
; Interrupt-Empfang mit eigenem Ringpuffer
; direkter Programmierung des UART (serielle Schnittstelle)
; Komar WS03

CODE    SEGMENT
        .486
        ASSUME  CS:CODE, DS:CODE

      ORG     00H
;KOPF DES TREIBERS
      DW      -1,-1           ; FFFF:FFFF
      DW      0C000H          ; Graeteattribut
      DW      OFFSET STRAT
      DW      OFFSET INTR
      DB      'DRVSER    '
FKT_TAB DW      OFFSET  INIT      ;Fktn 00 Initialisierung
      DW      OFFSET  DUMMY
      DW      OFFSET  DUMMY
      DW      OFFSET  LESEN     ;Fktn  03 IOCTL-Direktes Lesen
      DW      OFFSET  LESEN     ;Fktn. 04 Lesen aus ser.Puffer
      DW      OFFSET  ZEI_LES   ;Fktn. 05 Lesen ohne Entfernen
      DW      OFFSET  EMP_STAT  ;Zeichen verfuegbar? Empfang-Stat
      DW      OFFSET  LOESCH    ;Seriellen Empfangs-Ringpuffer loeschen
      DW      OFFSET  SCHREIB   ;Fktn  08 Zeichen seriell senden
      DW      OFFSET  DUMMY
      DW      OFFSET  SCHR_STA  ;Fktn. 0Ah Sendestatus pruefen
      DW      OFFSET  DUMMY
      DW      OFFSET  SCHREIB   ;Fktn. 0Ch Direktes Schreiben
      DW      OFFSET  DUMMY
      DW      OFFSET  DUMMY
      DW      OFFSET  DUMMY
      DW      OFFSET  DUMMY
      DW      OFFSET  DUMMY     ;INTR-SCH ;Fktn. 10h Spooler-Ausgabe
      DW      OFFSET  NO_SUP
      DW      OFFSET  NO_SUP
      DW      OFFSET  NO_SUP
      DW      OFFSET  NO_SUP    ;Verboten , nicht verfügbar
      DW      OFFSET  NO_SUP
      DW      OFFSET  NO_SUP
      DW      OFFSET  NO_SUP
;
DB_PTR     DW      (?),(?)
INMELD     DB 'Serieller Schnittstellen-Treiber DRVSER.SYS $'
FSEND      db "    Sendefehler-serielle Schnittstelle    $"
FEMPF      db "    Empfangsfehler-serielle Schnittstelle   $"
FSERI      db "    Fehler-serielle Schnittstelle $"
FEHLMEL    DB      'FEHLER$'
COMxALT    DW      2 DUP(0)
loob       db      0h ; LOOPBACK ein=10h;  0h = aus
comx       dw      0  ; 0 = COM1    1 = COM2
fifo       db      7  ; FIFO ein    0 = FIFO aus
portadr    dw      03F8h ;COM1
irqfrei    db      0EFh  ;Freigabe fuer COM1
irqsper    db      10h   ;Sperre   fuer COM1 - 08h fuer COM2
irqold     db      (?)
vektor     db      0Ch   ;Int.-Vektor fuer COM1- 0Bh fuer COM2
```

```
;
;ROUTINEN UND FUNKTIONEN DES TREIBERS
;
STRAT PROC    FAR
      MOV     CS:DB_PTR,BX       ; Adresse des Datenblocks ( ES:BX)
      MOV     CS:DB_PTR+2,ES     ; in DB_PTR merken
      RET
STRAT ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
INTR   PROC    FAR          ; DOS-Aufruf zur Aktivierung einer der
       PUSHF               ; Treiberfunktionen
       PUSHA
       PUSH    DS
       PUSH    ES
;
       PUSH    CS
       POP     DS
;
       LES     DI,DWORD PTR DB_PTR  ; Datenblockadresse nach ES:DI
       MOV     BL,ES:[DI+2]         ; Funktionsnummer ( Befehlscode)
       CMP     BL,24                ; holen
       JLE     BC_OK
       MOV     AX,8003h             ; Fehlercode ' Unbekannter Befehl'
       JMP     SHORT INTR_END       ; zurück
BC_OK: SHL     BL,1                 ; Zeiger fuer Sprungtabelle errechnen
       XOR     BH,BH
       CALL    [FKT_TAB+BX]         ; Funktion aufrufen
       LES     DI,DWORD PTR DB_PTR  ; Datenblockadresse nach ES:DI

INTR_END: OR     AX,100h              ; Fertig-Bit nach
          MOV    ES:[DI+3],AX         ; Statuswort

       POP   ES
       POP   DS
       POPA
       POPF
       RET
INTR   ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DUMMY   PROC   NEAR
        XOR    AX,AX      ; alles o. k.   bewirkt nichts
        RET               ; zurück zum Aufrufer
DUMMY   ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
NO_SUP    PROC   NEAR     ;fuer Funktionen, die eigentlich nicht aufgerufen
          MOV    AX,8003h ;werden duerfen
          RET             ;Fehler:  Befehl nicht erkannt
NO_SUP    ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

LESEN   PROC
        push    dx
        push    ds
        push    es
        push    di
        push    si

        mov     dx,0
        mov     cx,es:[di+12h]  ;Anzahl zu lesender Zeichen
        jcxz    readen

        lds     si,dword ptr es:[di+0Eh] ;DS:SI DOS-Puffer
```

```
????????????????????????????????????????????????????????????????
?
?
?
?
?
?
?
?
?
?
????????????????????????????????????????????????????????????????

readen: mov     word ptr es:[di+12h],dx ;Anzahl gelesener Zeichen
        xor     ax,ax

        pop     si
        pop     di
        pop     es
        pop     ds
        pop     dx
        ret

FEHLZEI: MOV     AX,800Bh    ; Lesefehler, wenn fehlerhafter Empfang

        pop     si
        pop     di
        pop     es
        pop     ds
        pop     dx
        ret

LESEN   ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
EMP_STAT PROC   NEAR            ;Empfangsstatus pruefen
         PUSH   DI
         XOR    AX,AX           ;Zeichen empfangen
         CLI
         MOV    DI,CS:SERNEXT   ;naechstes Zeichen = Lesezeiger
         CMP    DI,CS:SERLAST   ;Lese- = Schreibzeiger ?
         STI
         JNE    BEEND           ;wenn gleich -> Puffer leer
         MOV    AX,0200h        ;Fehler Treiber beschaeftigt-Puffer leer
BEEND:   POP    DI
         RET
EMP_STAT ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
LOESCH   PROC   NEAR         ; Empfangs-Ringpuffer leeren/loeschen
         PUSH   DI
         mov    di,offset SERBUF
         mov    cs:SERNEXT,di     ;Ringpuffer leeren
         mov    cs:SERLAST,di
         POP   DI
         RET
LOESCH   ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ZEI_LES  PROC
         PUSH   DS           ;naechstes Zeichen aus Ringpuffer in AL
         PUSH   SI           ;ausgeben, aber nicht entfernen
         PUSH   ES
         PUSH   DI

         CLI
```

```
          MOV  DI,CS:SERNEXT   ;naechstes Zeichen = Lesezeiger
          CMP  DI,CS:SERLAST   ;Lese- = Schreibzeiger ?
          JNE  NOTINH          ;wenn gleich -> Puffer leer
          POP  DI
          MOV  AX,0200h        ;Fehler Treiber beschaeftigt
          jmp  zfini

NOTINH:   MOV     AX,CS:[DI]   ;Zeichen aus Ring-Puffer nach AX

          cmp     ax,0FFFFh   ; Fehler Interrupt
          je      FEHLZE      ;
          test    ah,00011110b ;Empfangs-Fehler ?
          jnz     FEHLZE      ; ja,

          POP  DI
          LES  DI,DWORD PTR CS:[DB_PTR]
          LDS  SI,DWORD PTR ES:[DI+0EH]
          MOV  DS:[SI],AL             ;Zeichen in Uebergabepuffer
          XOR  AX,AX                  ;ALLES OK
          jmp  zfini

FEHLZE:   POP  DI
          MOV  AX,800Bh     ; Lesefehler

zfini:    STI
          POP  ES
          POP  SI
          POP  DS
          RET
ZEI_LES   ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SCHREIB    PROC       ;Schreiben/Senden nach Datei/Schnittstelle
           pusha
           push  es
           push  ds

           MOV   CX,WORD PTR ES:[DI+12H]  ; Anzahl der zu sendenden Zeichen
           jcxz  SCHEND                   ; kein Zeichen ?

           LDS   SI,DWORD PTR ES:[DI+0EH] ; im PUFFER ( Uebergabe )

????????????????????????????????????????????????????????????????????????????
?
?
?
?
?
????????????????????????????????????????????????????????????????????????????

SCHEND:    pop   ds
           pop   es
           popa
           XOR   AX,AX                    ;alles o.k.
           RET

SENDFEHL:  pop   ds
           pop   es
           popa
           MOV   AX,800Ah                ; Sendefehler-Code
           RET
SCHREIB    ENDP

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
SCHR_STA  PROC
          PUSH  ES                ; Abfrage des Status der  seriellen
                                  ; Schnittstelle bezüglich 'Senden'
          CALL  TESTEN            ; LSR in AH zurueck, AL unveraendert
          TEST  AH,08H            ; Break-Fehler
          JZ    NEXT
          MOV   AX,800Ch          ; allgemeiner Fehler
          JMP   EXRET

NEXT:     TEST  AL,20h
          JNZ   OKAY
          MOV   AX,0200h          ; Fehler  Sender beschaeftigt
          JMP   EXRET

OKAY:     XOR   AX,AX

EXRET:    POP   ES
          RET
SCHR_STA  ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
INIT    PROC    NEAR
        pusha
        push    ES
        push    DS

        MOV     WORD PTR  ES:[DI+14],OFFSET SEREND+2
        MOV     ES:[DI+14+2],CS
;
        CLI                     ; Interrupts sperren
        MOV     DX,cs:comx       ;Nummer der Schnittstelle COM1/COM2
        CALL    SERINIT          ;Initialisierungsroutine
;
        IN      AL,21H       ;IRQ4 im Interrupt-Controller freigeben
        MOV     irqold,AL    ;alten Int.-Master-Status retten
        AND     AL,irqfrei   ;
        OUT     21H,AL

        MOV     AH,35h        ; alten Interrupt-Vektor retten
        MOV     AL,vektor     ; fuer .sys-Treiber eigentlich
        INT     21H           ; nicht notwendig
        MOV     COMxALT,BX
        MOV     COMxALT+2,ES

;neuen Interrupt-Vektor in Vektor-Tabelle schreiben

        MOV     AH,25h         ;neuer Interrupt-Vektor fuer
        MOV     AL,vektor
        MOV     DX,OFFSET INTSER ;serielle Empfangsroutine
        INT     21H
        STI                     ; Interrupts zulassen

        MOV     AH,09
        MOV     DX,OFFSET INMELD
        INT     21h
;
        pop     DS
        pop     ES
        popa

        XOR     AX,AX

        RET
INIT    ENDP
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SERINIT PROC    NEAR        ;in DX die Schnittstellennummer
        PUSHA               ;uebergeben -9600Bd- gerade Paritaet-
        push    ES
        push    DS

        MOV     AX,40H      ;-8 Datenbit - 2 Stoppbit
        MOV     ES,AX       ;Bios-Daten adressieren
        MOV     BX,DX
        SHL     BX,1
        MOV     DX,ES:[BX]     ;Basisadresse der 8250-Register holen
        MOV     CS:portadr,DX  ;Basisadresse speichern

????????????????????????????????????????????????????????????????????????????
?
?
?
?
?
?
?
?
?
?
?
?
?
?
?
?
?
?
?
?
?
?
?
?
?
?
????????????????????????????????????????????????????????????????????????????

        STI                 ;Interrupts zulassen
;
        SUB     DX,4        ;Basis-Port 0
        IN      AL,DX       ;Empf.Puffer leeren

        ADD     DX,5        ;Line-Status-Register
        IN      AL,DX       ;Fehlerbit im Statusreg. loeschen

        sub    dx,3         ;FIFO Control Register
        mov    al,fifo      ;Sende- und Empfangspuffer ein/aus
        out    dx,al

        mov    di,offset SERBUF
        mov    cs:SERNEXT,di     ;Ringpuffer leeren
        mov    cs:SERLAST,di

        pop    DS
        pop    ES
        POPA
        RET
SERINIT ENDP
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;empfangenes Zeichen nach Low-Byte, LSR-Status nach High-Byte
;gesamtes Word in Ringpuffer schreiben
;bei Fehler-Interrupt FFFFh im Ringpuffer ablegen

INTSER  PROC   FAR        ; ISR fuer seriellen Empfang
        PUSHA             ; mit Ringpuffer
        PUSH   ES
        PUSH   DS

        PUSH   CS
        POP    DS
        mov    dx,cs:portadr   ;Basisadresse von COMx
        ADD    DX,2            ;Interrupt-Identifizierungsreg.
        IN     AL,DX
        AND    AL,07h           ;Interrupt anhaengig Bit.0 = 0
        CMP    AL,04h
        JNE    FEHLEM           ;Empfangsdaten eingetroffen ?

????????????????????????????????????????????????????????????????????
?
?
?
?
?
?
?
?
?
?
?
?
?
?
?
?
?
?
?
?
?
????????????????????????????????????????????????????????????????????

        POP    DS
        POP    ES
        POPA
        IRET
INTSER  ENDP
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
TESTEN  PROC   NEAR         ; LSR in AH uebergeben, AL unveraendert
        pushf
        push   dx
        push   bx
        mov    bl,al

        mov    dx,CS:portadr
        ADD    DX,5          ;ser. Statusregister adressieren
        IN     AL,DX         ;Status einlesen
        mov    ah,al

        mov    al,bl
        pop    bx
        pop    dx
        popf
        RET
TESTEN  ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SENDxx  PROC   NEAR          ;Senden eines in AL uebergebenen
        PUSHA                ;Datenbytes
        mov    dx,cs:portadr
        OUT    DX,AL         ;Datenbyte senden
        POPA
        RET
SENDxx  ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SENDEN  PROC   NEAR          ;Senden eines in AL uebergebenen
        PUSH   AX            ;Datenbytes

WARTSx: CALL   TESTEN        ;LSR-Status in AH
        TEST   AH,10h        ;Fehlerbit gesetzt ????? 10h ????
        JNZ    FALSES
        TEST   AH,20h        ;Sendepufferregister leer ?
        JZ     WARTSx        ;nein, dann warten

        CALL   SENDxx        ; AL senden
        CLC                  ; Carry = 0 kein Fehler
        POP    AX
        RET

FALSES: STC                  ;Carry = 1 Fehler
        POP    AX
        RET
SENDEN  ENDP
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
BUFWRITE PROC  NEAR     ;AX-Inhalt in seriellen Ring-Puffer
         PUSHF          ;schreiben
         PUSHA
         CLI

??????????????????????????????????????????????????????????????????????
?
?
?
?
?
?
?
?
?
????????????????????????????????????????????????????????????????????????

         STI                      ;Interrupts freigeben
         POPA
         POPF
         RET
BUFWRITE ENDP
;
BUFLES  PROC NEAR         ;naechstes Byte aus Ringpuffer in AX
        PUSHF             ;ausgeben
        PUSH  DI
        CLI
        MOV  DI,CS:SERNEXT   ;naechstes Zeichen = Lesezeiger
        CMP  DI,CS:SERLAST   ;Lese- = Schreibzeiger ?
        JNE  NOTLER          ;wenn gleich -> Puffer leer
        STI
        POP  DI
        POPF
        STC                  ;CY = 1 ->  Puffer leer
        RET

NOTLER: MOV  AX,CS:[DI]   ;Zeichen aus Puffer nach AX
        INC  DI
        INC  DI
        CMP  DI,OFFSET SEREND ;Ende des Ringpuffers erreicht ?
        JNE  EXR
        MOV  DI,OFFSET SERBUF ;Lesezeiger auf Pufferanfang setzen
EXR:    MOV  CS:SERNEXT,DI
        STI
        POP  DI
        POPF
        CLC                   ;CY = 0 -> Pufferinhalt in AX
        RET
BUFLES  ENDP
;
SERNEXT DW   OFFSET SERBUF
SERLAST DW   OFFSET SERBUF
SERBUF  DW   2000 DUP (0)
SEREND  EQU  THIS BYTE
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
CODE    ENDS
        END
```

Anhang 2: DRVTERM3.asm

```
;DRVTERM3.asm Treibertest fuer DRVSER.SYS  Terminalemulation mit serieller
;Schnittstelle mit DOS-Handle und DOS-IOCTL-Funktionen    Komar  WS03
;TASM DRVTERM3    TLINK /t DRVTERM3+eados
;greift ueber installierten DOS-Geraetetreiber DRVSER.SYS mit DOS-Fktn.
;auf serielle Schnittstelle zu - Beendigung mit Strg C
;DOS-Treiber DRVSER.sys muss installiert sein

        INCLUDE     eados.inc

       .MODEL TINY
       .486
       .CODE
       ORG      100H

START: MOV      AX,3D02H            ;Datei oeffnen-lesen und schreiben
       MOV      DX,OFFSET DATNAME   ;Datei ist hier  der Serielle Treiber
       INT      21H                 ;DRVSER
       JNC      OK
       or       ax,0D000h           ;Fehlerkennung
       call     haxaus
       JMP      FEHLER              ;Fehlermeldung und beenden
;
OK:    MOV      HANDLE,AX           ;geliefertes handle abspeichern
;
inilop: mov      ax,3F00h      ;Empfangs-Ringpuffer loeschen
        mov      bx,handle     ;durch sukzessives Lesen
        mov      cx,1
        mov      dx,offset epuffer
        int      21h
        jc       warten
        cmp      ax,0           ;kein Zeichen gelesen ?
        jne      inilop

WARTEN: MOV      AX,4406H      ;Geraetetreiber: Eingabestatus abfragen
        MOV      BX,HANDLE
        INT      21H
        JNC      ABFR
        or       AX,0C000h
        call     haxaus
        JMP      FEHLER

ABFR:   CMP      AX,00
        JE       SEND1            ;Treiber nicht bereit zum Empfang

        MOV      AX,4402H  ;3F00h Daten von einem Zeichentreiber empfangen
        MOV      BX,HANDLE
        MOV      CX,1             ; 1 Zeichen holen
        MOV      DX,OFFSET EPUFFER
        INT      21H
        JNC      EMPF
        or       ax,8000h
        call     haxaus
        JMP      FEHLER

EMPF:   CMP      AX,0000
        JE       FEHLER
        MOV      AL,EPUFFER      ; aus Empfangs-Puffer
        MOV      AH,0EH          ; empfangenes Zeichen auf Bildschirm aus
        INT      10H
;
```

```
SEND1:  MOV     AX,4407h        ; Ausgabestatus abfragen
        MOV     BX,HANDLE
        INT     21H
        JNC     LP0
        or      ax,7000h
        call    haxaus
        JMP     FEHLER

LP0:    CMP     AX,0000         ; Treiber nicht bereit
        JNE     LP1
        JMP     WARTEN

LP1:    MOV     AH,1            ; Zeichen von Tastatur ?
        INT     16H
        JZ      WARTEN
        MOV     AH,0
        INT     16H             ; Zeichen von Tastatur einlesen
        CMP     AL,03H          ; Strg C = Beenden
        JE      EXIT

        MOV     SPUFFER,AL       ; Zeichen nach Sendepuffer-Puffer

        MOV     AX,4403H ;4000h Datei beschreiben - Daten an Zeichentr.
        MOV     BX,HANDLE
        MOV     CX,1                ;Anzahl Zeichen
        MOV     DX,OFFSET SPUFFER
        INT     21H
        JNC     WARTEN
        or      ax,0F000h
        call    haxaus
;
FEHLER: MOV     AH,09H
        MOV     DX,OFFSET FEHLMEL
        INT     21H

EXIT:   MOV     AX,3E00H         ;geoeffnete Datei schliessen
        MOV     BX,HANDLE
        INT     21H              ;funktioniert nicht
        jnc     fini

        MOV     AH,09H
        MOV     DX,OFFSET FEHL3E
        INT     21H

fini:   MOV     AH,4CH
        INT     21H

FEHLMEL DB      'FEHLER$'
FEHL3E  DB      10,13,' Fehler Fktn 3Eh $'
;
HANDLE   DW  (?)
DATNAME  DB  'DRVSER.SYS',00
EPUFFER  DB  16 DUP('$')                ;Empfangspuffer
SPUFFER  DB  16 DUP('$')                ;Sendepuffer
         END  START
```