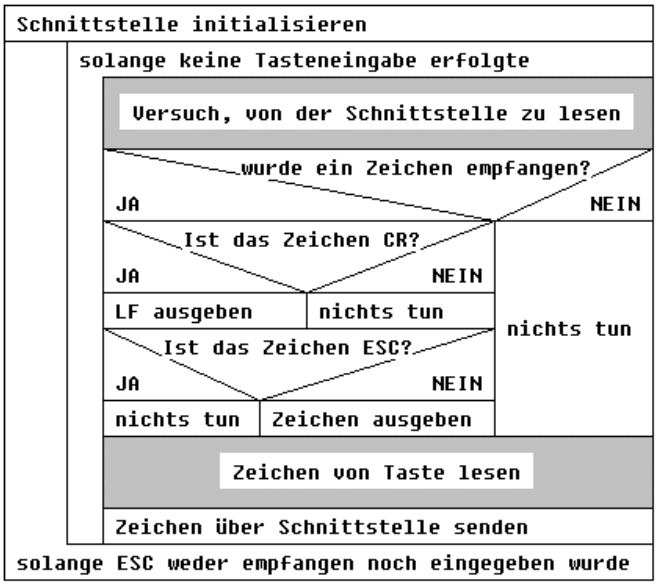
Assembler wieder in?

Mängel der systemnahen C-Funktionen beim Programmieren der seriellen Schnittstelle

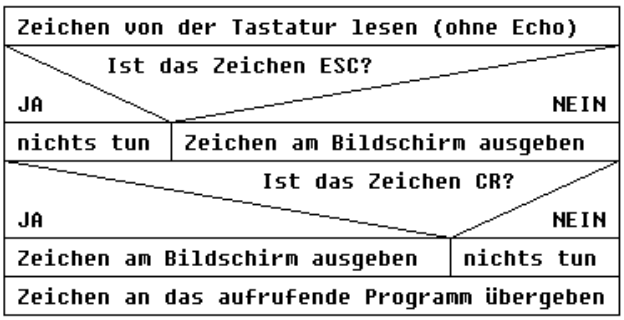
Walter Riemer DSK-472: SER.ZIP, BNU202.ARJ

Im Unterricht wurde die Aufgabe gestellt, mittels eines Terminal-Programms den zeichenweisen Verkehr zwischen zwei Rechnern in beiden Richtungen zu ermöglichen, die über ein Nullmodem-Kabel verbunden sind. Im Grundausbau soll nur die Kommunikation in beiden Richtungen möglich sein, beidseitige Beendigung erfolgt durch ESC-Eingabe auf einem der beiden Rechner. Im weiteren Ausbau sind beliebige Erweiterungen in Richtung komfortablerer Bedienung u.dgl. möglich.

Das Programm sollte zum Beispiel laut folgendem dreiteiligen Struktogramm modular aufgebaut sein.



Zeichen von Taste lesen



Das Programm wurde ursprünglich unter Verwendung der Borland-C-eigenen Funktion `bioscom()` geschrieben (Variante SerTerm1.C). Auf den Rechnern (486-DX2-Tower und 486-SL-Notebook) des Entwicklers lief diese Fassung einwandfrei, nicht jedoch zwischen dem Notebook und den meisten Arbeitsplatz- Rechnern in TGM-Computersälen.

Aufgrund von Gesprächen mit Kollegen, wonach die `bioscom`-Funktion bekanntermaßen nicht verläßlich sei, wurde das Programm geändert (Variante `SerTest1.C`). Sämtliche `bioscom`-Aufrufe wurden durch entsprechende `int86()`-Aufrufe ersetzt. Diese Variante funktionierte auch zwischen Notebook und den Schulrechnern, nicht jedoch zwischen zwei Schulrechnern, insgesamt also etwas besser, aber nicht verläßlich.

Um sicher zu gehen, wurde das Programm daraufhin zur Gänze in Assemblersprache geschrieben (Variante `SerAsm.ASM`, funktionsgleich), wobei die serielle Schnittstelle nicht mittels INT 14h, sondern direkt angesprochen wurde. Dieses Programm funktionierte auf den Rechnern des Entwicklers und auf den Schulrechnern.

Schließlich wurden die `int86`-Aufrufe in Anpassung an die Assembler-Version mittels `inport-` und `outport`-Funktionen in C geschrieben (Variante `SerTestA.C`); nur die Initialisierungsroutine wurde mittels `biscom()` belassen, da sie unkritisch erschien.

Diese Assembler-nahe Fassung war in der Handhabung der für die Übertragung maßgebenden Variablen sogar viel einfacher als die mit `bioscom()`, da sie alle `char`-Variablen sind. Erwähnenswert ist vielleicht, daß die C-.EXE-Dateien ungefähr 30 kB groß sind (wovon allerdings ein großer Teil auf Konto der `printf()`-Funktion geht), während das Assembler-.COM-File 218 Bytes umfaßt. Die Quellprogramme sind sind alle etwas über 3 kB groß und umfassen 82 Zeilen (die .ASM-Fassung) bzw. zwischen 96 und 109 Zeilen (die .C-Versionen).

Mit diesen Erfahrungen kann wieder einmal eine Lanze für das Assembler-Programmieren gebrochen werden: in gewissen Bereichen ist das Assemblerprogrammieren nach wie vor unentbehrlich.

Nachträglich wurde noch einem Hinweis von Koll. Fiala nachgegangen: Bei allen Übertragungsprogrammen über die serielle Schnittstelle erweist sich der im PC ursprünglich sehr sparsam und ungepuffert programmierte Interrupt 14h (zusammen mit 0Bh und 0Ch) als ungeeignet und wird immer durch geignete Treiber ersetzt. Nur bei PS/2- und EISA-Rechnern ist die BIOS-Funktion in dieser Hinsicht korrekt programmiert, sie brauchen daher derartige Treiber nicht, was allerdings ein schwacher Trost ist, da unsere Terminal-Programme ja auf allen Rechnern laufen sollen.

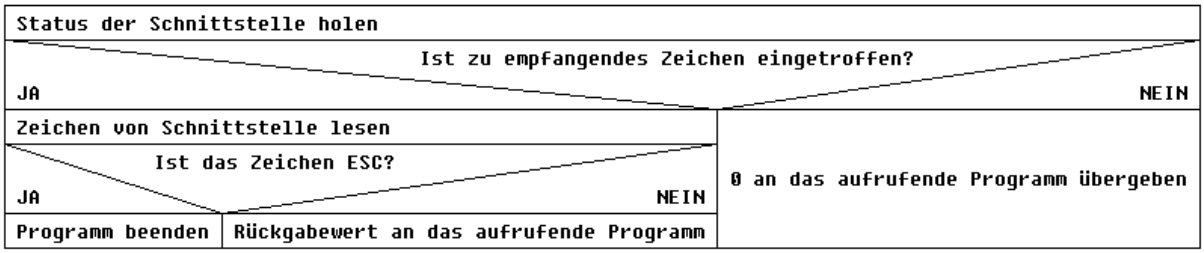
Ein Treiber dieser Art, der resident installiert werden kann und dann die ungepuffert arbeitende BIOS- Routine 14h ersetzt, heißt `BNU.COM`. Dieser Treiber kann durch Kommandozeilenschalter vielfältig gesteuert werden (Information darüber erhält man mit `BNU /?`); er kann auch wieder aus dem Speicher entfernt werden (`BNU /U`).

Siehe da: sobald BNU resident geladen war, funktionierte auch die Urversion (`SerTerm1.C`) auf den Schulrechnern problemlos, erst recht natürlich die verbesserten Versionen. Wissen muß man's halt, und der Firma Borland kann man den Vorwurf nicht ersparen, daß sie sich über die im Zusammenhang mit `bioscom()` möglichen Kalamitäten zum Schaden der Anwender ausschweigt.

Es folgen die Quellprogramm-Listings:

□

Versuch, von der Schnittstelle zu lesen



1. SerTerm1.C, mit bioscom (die "Ur-Variante")

```
/* SerTerm1.C: Terminal-Programm mittels COM1 */
#include <stdlib.h>
#include <bios.h>
#include <math.h>
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <dos.h>

#define ESC 0x1B

char TastLes(void);
void Senden(char Zeichen);
unsigned int Horchen(void);

enum {COM1, COM2};

void InitCom (int ComPort, /* +-----------------------------+ */
              int Baud,    /* | Der Initialisierungsparameter| */
              int DataBits,/* | ist ein Byte-Wert, der zur   | */
              int StopBits,/* | leichteren Handhabung aus    | */
              char Parity) /* | "sprechenden" Parametern     | */
{ char Paritaet;           /* | zusammengesetzt wird.        | */
  switch (Parity)          /* |                              | */
  { case 'N': Paritaet=0;  /* | Bits Bedeutung               | */
              break;       /* | 7-5  Baudrate (100 = 1200)   | */
    case 'O': Paritaet=1;  /* | 4,3  Parität  (0 = keine)    | */
              break;       /* |   2  Stopbits (1 = zwei)     | */
    case 'E': Paritaet=3;  /* | 1,0  Zeichenlänge (11 = 8)   | */
  }                        /* +-----------------------------+ */
  bioscom(0,                        /* Initialisieren */
          (1+(int) ceil(log(Baud/150)/log(2))) << 5
           | Paritaet << 3
           | (StopBits-1) << 2
           | DataBits-5,             /* Initialisierungsparameter */
          ComPort);                  /* Auswahl COM1 oder COM2    */
}

char TastLes(void)
{ char Zeichen;
  Zeichen=getch();
  if (Zeichen != ESC)
     putch(Zeichen);
  if (Zeichen == 0x0D)
     putch(0x0A);
  return Zeichen;
}

void Senden(char Zeichen)
{ bioscom (1,     /* auf Schnittstelle schreiben */
           Zeichen,
           COM1);  /* Auswahl COM1 oder COM2    */
}

unsigned int Horchen(void)
{ int RueckWert;
  char Zeichen;
  RueckWert=bioscom (3,0,COM1);
  if (RueckWert & 0x0100)          /* Zeichen zum Lesen da ?  */
     { RueckWert = bioscom (2,     /* Ja: von Schnittstelle lesen  */
                            0,     /* Inhalt beliebig           */
                            COM1); /* Auswahl COM1 oder COM2    */
       if ((RueckWert & ESC) == ESC)  /* wurde ESC empfangen ? */
       { puts("\nProgrammlauf durch Empfang von ESC beendet.\n");
         getch();
         exit(0);
       }
       if ((RueckWert & 0x8000) == 0) /* kein Time-out ? */
          return RueckWert;
     }
  else
     return 0;
}

main()
{ unsigned int RueckWert;
  char Zeichen;
  InitCom(COM1,1200,8,2,'N');
  clrscr();
  puts("");
  do
  { while (!kbhit())
    { RueckWert=Horchen();  /* Leseversuch auf Schnittstelle */
      if (RueckWert != 0)   /* Zeichen da?                   */
      { if ((RueckWert & 0x00FF) == 0x000D)  /* Ja: CR ?     */
           putchar(0x0A);   /* Ja: LF ausgeben               */
        if ((RueckWert != ESC) && (RueckWert != 0x8000))
           putchar (RueckWert);
      }
    }
    Zeichen=TastLes();
    Senden(Zeichen);
  }
  while ((RueckWert != ESC) && (Zeichen != ESC));
  puts("\nProgrammlauf durch Eingabe von ESC beendet.\n");
  getch();
}
```

2. SerTest1.C, mit int86-Funktion

```
/* SerTest1.C: Terminal-Programm, feste Parameter, ohne bioscom */
#include <stdlib.h>
#include <bios.h>
#include <math.h>
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <dos.h>
#include <ctype.h>

#define ESC 0x1B

union REGS Register;
int ComPort=0, Baud=1200, DataBits=8, StopBits=2, Parity='N';

char TastLes(void);
void Senden(char Zeichen);
unsigned int Horchen(void);

void InitCom (int ComPort,  /* +-----------------------------+ */
              int Baud,     /* | Der Initialisierungsparameter| */
              int DataBits, /* | ist ein Byte-Wert, der zur   | */
              int StopBits, /* | leichteren Handhabung aus    | */
              char Parity)  /* | "sprechenden" Parametern     | */
{ char Paritaet;            /* | zusammengesetzt wird.        | */
  switch (toupper(Parity))  /* |                              | */
  { case 'N': Paritaet=0;   /* | Bits Bedeutung               | */
              break;        /* | 7-5  Baudrate (100 = 1200)   | */
    case 'O': Paritaet=1;   /* | 4,3  Parität  (0 = keine)    | */
              break;        /* |   2  Stopbits (1 = zwei)     | */
    case 'E': Paritaet=3;   /* | 1,0  Zeichenlänge (11 = 8)   | */
  }                         /* +-----------------------------+ */
  Register.h.ah=0;
  Register.h.al=(1+(int) ceil(log(Baud/150)/log(2))) << 5
                | Paritaet << 3
                | (StopBits-1) << 2
                | DataBits-5;            /* Initialisierungsparameter */
  Register.x.dx=ComPort;
  int86(0x14,&Register,&Register);
}

char TastLes(void)
{ char Zeichen;
  Zeichen=getch();
  if (Zeichen != ESC)
     putch(Zeichen);
  if (Zeichen == 0x0D)
     putch(0x0A);
  return Zeichen;
}

void Senden(char Zeichen)      /* auf Schnittstelle schreiben */
{ Register.h.ah=1;
  Register.h.al=Zeichen;
  Register.x.dx=ComPort;
  int86(0x14,&Register,&Register);
}

unsigned int Horchen(void)
{ int RueckWert;
  char Zeichen;
  Register.h.ah=3;
  Register.x.dx=ComPort;
  RueckWert=int86(0x14,&Register,&Register);
  if (RueckWert & 0x0100)          /* Zeichen zum Lesen da ?  */
  { Register.h.ah=2;
    Register.x.dx=ComPort;
    RueckWert=int86(0x14,&Register,&Register);
    if ((RueckWert & ESC) == ESC)  /* wurde ESC empfangen ? */
    { puts("\nProgrammlauf durch Empfang von ESC beendet.\n");
      getch();
      exit(0);
    }
    if ((RueckWert & 0x8000) == 0) /* kein Time-out ? */
       return RueckWert;
  }
  else
    return 0;
}

main()
{ unsigned int RueckWert;
  char Zeichen;
  InitCom(ComPort,Baud,DataBits,StopBits,Parity);
  delay(100);
  clrscr();
  puts("");
  do
  { while (!kbhit())
    { RueckWert=Horchen();  /* Leseversuch auf Schnittstelle */
      if (RueckWert != 0)   /* Zeichen da?                   */
      { if ((RueckWert & 0x00FF) == 0x000D)  /* Ja: CR ?     */
           putchar(0x0A);   /* Ja: LF ausgeben               */
        if (RueckWert != ESC)
           putchar (RueckWert);
      }
     }
     Zeichen=TastLes();
     Senden(Zeichen);
  }
  while ((RueckWert != ESC) && (Zeichen != ESC));
  puts("\nProgrammlauf durch Eingabe von ESC beendet.\n");
  getch();
}
```

3. SerAsm.ASM, Assemblerprogramm

```
          TITLE    "SerAsm.ASM: Terminalprogramm, COM1,
Standardparameter"
CodeSeg   SEGMENT  PARA PUBLIC 'Code'
          ASSUME   CS:CodeSeg,SS:CodeSeg
          ASSUME   DS: CodeSeg,ES:CodeSeg
ESCP      EQU      1Bh
LF        EQU      0Ah
CR        EQU      0Dh
          ORG      100h
Begin:    JMP      Start
ESCEmpfMsg DB      LF,CR,"Programmlauf durch Empfang von ESC
beendet.",'$'
EscEingMsg DB      LF,CR,"Programmlauf durch Eingabe von ESC
beendet.",'$'
Start:    CALL     ClrScr      ; Bildschirm löschen
          CALL     SetModReg   ; Modem-Kontrollregister setzen
; Horchschleife: solange kein Tastendruck, Leseversuch vom COMx
HorchSchl: MOV     AH,1        ; nein: BIOS-Tastaturabfrage
          INT      16h         ; erfolgte Tastendruck ?
          JZ       HSHorch     ; nein: Horchen, ob Zeichen auf COMx
          CALL     TastLes     ; ja: eingegebenes Zeichen lesen und
ausgeben
          CALL     Senden      ;     Zeichen über COMx senden
          JMP      HorchSchl
HSHorch:  CALL     Horchen
          CMP      AL,0        ; Zeichen da ?
          JE       HorchSchl   ; nein
          CMP      AL,ESCP     ; ja: ESC empfangen ?
          JE       HSExit      ;     ja: Programm beenden
          CALL     ALAusgeb    ;     nein: ausgeben
          JMP      HorchSchl
HSExit:   LEA      DX,EscEmpfMsg
Exit:     MOV      AH,9        ; Zeichenkette ausgeben
          INT      21h
          MOV      AH,4Ch      ; Exit
          INT      21h
;
Horchen:  MOV      DX,3FDh     ; Leitungs-Statusregister
          IN       AL,DX       ; Status lesen
          TEST     AL,1        ; Bit 0 testen: Zeichen da ?
          JZ       HRetN       ; nein
          MOV      DX,3F8h     ; ja: Kanaladresse von COM1
          IN       AL,DX       ;     Zeichen lesen
          RET
HRetN:    MOV      AL,0        ; Kennung: kein Zeichen da
          RET
;
Senden:   MOV      DX,3F8h     ; Kanaladresse von COM1
          OUT      DX,AL       ; Zeichen schreiben
          RET
;
; TastLes: Zeichen von Tastatur lesen,
; am Bildschirm ausgeben (außer wenn ESC)
;         Wenn CR, dann LF zusätzlich ausgeben.
;         Letztes Zeichen in AL übergeben.
TastLes:  MOV      AH,0        ; BIOS-Tastatureingabe lesen
          INT      16h         ; eingegebenes Zeichen nach AL
          CMP      AL,ESCP     ; ESC gelesen ?
          JNE      TL1         ; nein
          CALL     Senden      ; ja
          LEA      DX,EscEingMsg
          JMP      Exit
TL1:      CALL     ALAusgeb    ; nein:
                               ; Zeichen aus AL am Bildschirm ausgeben
          CMP      AL,CR       ; war das Zeichen Carriage Return ?
          JNE      TLRet       ; nein: Zeichen übergeben
          CALL     Senden
          MOV      AL,LF       ; ja: zusätzlich LF ausgeben
          CALL     ALAusgeb
TLRet:    RET
;
ALAusgeb: MOV      AH,2        ; Zeichen auf Bildschirm ausgeben
          MOV      DL,AL       ; auszugebendes Zeichen
          INT      21h
          RET
;
SetModReg: MOV     DX,3FCh     ; Modem-Kontrollregister
          MOV      AL,3        ; DTR und RTS setzen
          OUT      DX,AL
          RET
;
ClrScr:   MOV      AH,0        ; Bildschirm-Modus einstellen
          MOV      AL,3        ; 25x80 Textmodus
          INT      10h         ; BIOS-Video-IO
          RET
;
CodeSeg   ENDS
          END      Begin
```

4. SerTestA.C, mit inport- und outport-Funktionen

```
/* SerTestA.C: Terminal-Programm, feste Parameter, Assembler-nah */
#include <stdlib.h>
#include <bios.h>
#include <math.h>
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <dos.h>
#include <ctype.h>

#define ESC 0x1B

enum {COM1, COM2};

unsigned int Port[]={0x3F8,0x2F8};

char TastLes(void);
void Senden(unsigned char Zeichen);
unsigned char Horchen(void);



void InitCom (int COM1,     /* +------------------------------+ */
              int Baud,     /* | Der Initialisierungsparameter| */
              int DataBits, /* | ist ein Byte-Wert, der zur   | */
              int StopBits, /* | leichteren Handhabung aus    | */
              char Parity)  /* | "sprechenden" Parametern     | */
{ char Paritaet;            /* | zusammengesetzt wird.        | */
  switch (toupper(Parity))  /* |                              | */
  { case 'N': Paritaet=0;   /* | Bits Bedeutung               | */
              break;        /* | 7-5  Baudrate (100 = 1200)   | */
    case 'O': Paritaet=1;   /* | 4,3  Parität  (0 = keine)    | */
              break;        /* |   2  Stopbits (1 = zwei)     | */
    case 'E': Paritaet=3;   /* | 1,0  Zeichenlänge (11 = 8)   | */
  }                         /* +------------------------------+ */
  bioscom(0,                        /* Initialisieren */
          (1+(int) ceil(log(Baud/150)/log(2))) << 5
           | Paritaet << 3
           | (StopBits-1) << 2
           | DataBits-5,            /* Initialisierungsparameter */
          COM1);                    /* Auswahl COM1 oder COM2    */
}

char TastLes(void)
{ char Zeichen;
  Zeichen=getch();
  if (Zeichen != ESC)
     putch(Zeichen);
  if (Zeichen == 0x0D)
     putch(0x0A);
  return Zeichen;
}

void Senden(unsigned char Zeichen)      /* auf Schnittstelle schreiben */
{ outportb(Port[COM1],Zeichen);
}


unsigned char Horchen(void)
{ int RueckWert;
  char Zeichen, Byte;
  Byte=inportb(Port[COM1]+5); /* Status vom Leitungs-Statusreg. lesen */
  if (Byte & 0x01)           /* wenn Zeichen zum Lesen da */
  { Zeichen=inportb(Port[COM1]);  /* Zeichen lesen */
    if (Zeichen == ESC)      /* wurde ESC empfangen ? */
    { puts("\nProgrammlauf durch Empfang von ESC beendet.\n");
      getch();
      exit(0);
    }
    return Zeichen;
  }
  else
    return 0;
}

main()
{ unsigned char Zeichen, Empfangen, Parity;
  InitCom(COM1,1200,8,2,'N');
  delay(100);
  clrscr();
  puts("");
  do
  { while (!kbhit())
    { Empfangen=Horchen();  /* Leseversuch auf Schnittstelle */
      if (Empfangen != 0)   /* Zeichen da?                   */
      { if (Empfangen == 0x0D)  /* Ja: CR ?        */
           putchar(0x0A);       /* Ja: LF ausgeben */
        if (Empfangen != ESC)
           putchar (Empfangen);
      }
     }
     Zeichen=TastLes();
     Senden(Zeichen);
  }
  while ((Empfangen != ESC) && (Zeichen != ESC));
  puts("\nProgrammlauf durch Eingabe von ESC beendet.\n");
  getch();
}
```

□