CLUBINFO

42. Ausgabe

Kontaktadresse:
Club 80
Hartmut Obermann
Mozartring 23
Postfach 1430
8870 Gunzburg
Tel.: 08221/30248
BTX: 08221/30248
Anrufbeantworter &
FAX: 08221/33575

Inhaltsverzeichnis

Neues vom Vorstand

Wie immer möchte ich mich am Anfang des Infos kurz zu Wort melden. Diesmal mit einem sehr, leider aber auch mit einem weniger erfreulichen Thema. Fangen wir mit letzterem an.

I. Südländertreffen/Jahreshauptversammlung des CLUB 80
Wie schon in den vergangenen Jahren, würde ich auch zum Jahresanfang 1994 gerne wieder ein Regional-Treffen Süd hier im Raum Ulm organisieren. Leider wird mir persönlich das diesmal nicht möglich sein. Da ich mit Wirkung vom 01.04.94 nach Karlsruhe versetzt werde und bis dahin beruflich sehr stark eingespannt bin (mal ganz abgesehen von den Umzugsvorbereitungen), sehe ich mich außerstande, die Organisation des Südländertreffens zu übernehmen. Fraglich ist ebenfalls noch, ob ich 1994 in der Lage sein werde eine Jahreshauptversammlung zu veranstalten.

Gesucht wird also ein CLUB 80-Mitglied, das willens und in der Lage ist, ein Regionaltreffen Süd und/oder eine Jahreshauptversammlung zu organisieren. Dabei kommt es vor allem darauf an, eine Unterkunft passender Größe (Unterbringungsmöglichkeit für max. 10 - 15 Teilnehmer) ausfindig zu machen, die noch dazu einen Nebenraum für die gleiche Anzahl Rechner zur Verfügung stellen kann. Das ganze natürlich zu einem vernünftigen Preis und in einer möglichst günstigen Verkehrslage.

"Das klingt schwerer als es in Wirlichkeit ist" sagt meine Frau, die in den letzten Jahren für diese Dinge zuständig war. Wie die Teilnehmer der letzten Treffen bestätigen können, war ich ja nur auf dem Papier der Organisator des Treffens und maximal für die Tagesordnung zuständig.

Sollte sich ein CLUB 80-Mitglied, oder seine bessere Hälfte, in der Lage sehen und noch dazu willens sein, ein Treffen zu organisieren, so möge er sich bitte bei mir melden. Wenn möglich noch vor dem nächsten Redaktionsschluß, damit im Info 43 eventuell schon Näheres veröffentlicht werden kann.

II. Clubinfo
In diesem Info erscheinen unter anderem ein paar Artikel, die auf Anregungen aus der Fragebogenaktion, früheren Infos und Gesprächen mit Mitgliedern entstanden sind. Stellvertretend genannt sei z.B. Alexander Schmid, der eine Fortsetzung des PCX-Programms für CP/M-Rechner und eine Lösung für das ASCII-Konvertierungsproblem zwischen verschiedenen Rechnertypen beisteuert. Oder Gerald Schröder, der seine Diplomarbeit etwas umgearbeitet hat und versucht, uns in die Geheimnise von Compilern einzuweihen.

Beiden möchte ich an dieser Stelle recht herzlich dafür danken, daß sie auf meine mehr oder weniger direkten "Arbeitsaufträge" so prompt reagieren und ganz wesentlich zum Informationsgehalt der Infos beitragen! Gedankt sei natürlich auch allen anderen, hier nicht namentlich genannten Infoschreibern, die sich die Zeit nehmen uns ihre EDV-Kenntnisse näher zu bringen.

In diesem Sinne wünsche ich euch allen ein frohes Weihnachtsfest und ein gesundes, gutes und erfolgreiches Jahr 1994,

Neue FIDO-Nummer

Wie schon im letzten Info berichtet, gibt es nach der Zwangsregionalisierung zwei FIDO-Netze. Das neue, zwangsregionalisierte wird als FIDO-Light bezeichnet, während die Konterrevolutionäre ihr Netz unter dem Namen FIDO-Classic betreiben.

Der SysOp der BRAINSTROM (ehemals 2:241/7902), die jahrelang meine Heimatbox war, hat sich nun leider dazu entschlossen, ganz aus dem FIDO-Netz auszutreten. Dadurch habe ich die Mailbox und damit auch die FIDO-Nummer gewechselt.

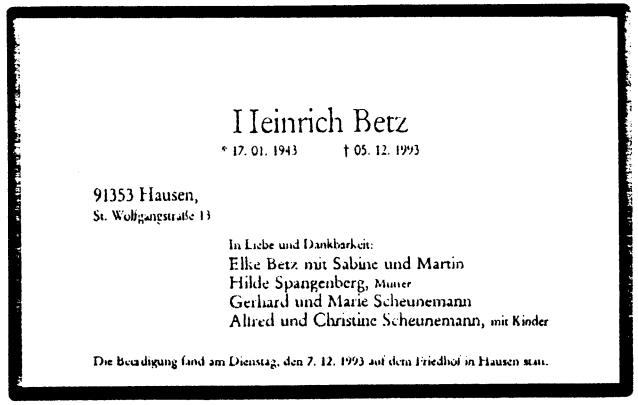
Ab sofort bin ich in der SYN-REL-Box (Tel.: 08282-4311) oder unter der FIDO-Nummer 2:241/7922.10 zu erreichen.

BTX-Anschluß

Durch die Probebestellung des kostenlosen 1&1-Modems ist der CLUB 80 jetzt auch per BTX zu erreichen. Leider ist eine Anmeldung im BTX nur unter dem Namen möglich, auf den auch der verwendete Telefonanschluß eingetragen ist. Aus diesem Grund ist der CLUB 80 unter Hartmut Obermann auf dem Anschluß **0822130248** zu erreichen.

Da ich allerdings BTX nicht regelmäßig nutze, kann es passieren, daß mich eine Nachricht längere Zeit nicht erreicht!

Wir trauern um einen guten
Freund und unser Clubmitglied



Heinrich Betz
* 17. 01. 1943 † 05. 12. 1993

91353 Hausen,
St. Wolfgangstraße 13

In Liebe und Dankbarkeit:
Elke Betz mit Sabine und Martin
Hilde Spangenberg, Mutter
Gerhard und Marie Scheunemann
Alfred und Christine Scheunemann, mit Kinder

Die Beerdigung fand am Dienstag, den 7. 12. 1993 auf dem Friedhof in Hausen statt.

Der Vorstand Club80

Termine... Termine... Termine... Termine... Termine

Graphic 1	Kassel	13.01.- 15.01.94
HTC Hamburger Computertage	Hamburg	20.01.- 23.01.94
Computerschau	Dortmund	02.02.- 06.02.94
Comdesign	Sinsheim	23.02.- 26.02.94
Cebit	Hannover	16.03.- 23.03.94

Redaktionsschluß für das nächste Clubinfo ist der

01. März 1994.

Bitte überprüft Eure Daten
und teilt uns Änderungen mit!

Beachtet die Umstellung der
Postleitzahlen!

Grafik des Genie IIIs unter Holte CP/M+

Volker Dose, Egbert Schröer, November 1993

Im ersten Beitrag zur kleinen Artikelserie über die Ansteuerung der Grafik des Genie IIIs unter CP/M wurde eine Möglichkeit in Assembler vorgestellt. Auch mit Hochsprachen wie Turbo Pascal ist dies möglich, wenn man dazu die BDOS-Funktion 50 nutzt. Die entsprechende Routine ist ebenso im Include GRAFIK.INC implementiert wie alle Funktionen zum Zeichnen/Löschen von Punkten, Linien etc.

```
(* INCLUDE-File GRAFIK.INC
   in das Programm einzufuegen mit

   ($I GRAFIK.INC)

  Es folgen einige Unterroutinen, mit denen die HRG des
  Genie IIIs angesprochen werden kann. Es sind folgende
  Funktionen, bzw Proceduren implementiert:

 Procedure  Gra_On (seite)
 Procedure  Gra_Off(seite)
 Procedure  Gra_Cls(farbe)

 Procedure  Set_Point(xKoor,yKoor,seite)
 Procedure  Res_Point(xKoor,yKoor,seite)
 Function   Ask_Point(xKoor,yKoor,seite)
 Procedure  Hrg_Line(xStart,yStart,xEnd,yEnd,farbe,seite)

 Procedure  Set_Point_Normiert(xkoor,ykoor,seite)
 Procedure  Res_Point_Normiert(xkoor,ykoor,seite)
 Procedure  Hrg_Line_Normiert(xStart,yStart,xEnd,yEnd,farbe,seite)
 Procedure  Hrg_Circle(xKoor,yKoor,radius,farbe,seite)

  Bei den nicht normierten Proceduren gilt

 xKoor, xStart und xEnd  :  INTEGER im Bereich von 0-639
 yKoor, yStart und yEnd  :  INTEGER im Bereich von 0-275
                         :  (25 Zeilen * 11 Scanlinien)
 seite                   :  BYTE im Bereich 0-1
 farbe                   :  Byte 0 = schwarz   1 = weiss

  Bei den normierten Proceduren gilt abweichend

 yKoor, yStart und yEnd  INTEGER im Bereich von 0-449


!  ACHTUNG ! ACHTUNG ! ACHTUNG ! ACHTUNG ! ACHTUNG ! ACHTUNG   !
!    Es wird nicht ueberprueft, ob die uebergebenen Parameter  !
!    im oben angegebenen Breich liegen. Eventuell kann der     !
!    Rechner bei einigen Proceduren abstuerzen.                !
!  ACHTUNG ! ACHTUNG ! ACHTUNG ! ACHTUNG ! ACHTUNG ! ACHTUNG   !

*)
(* ---------------------------------------- *)
(* Direkter BIOS-Aufruf ueber BDOS-Funktion 50 *)
(* geklaut aus CPM-IBM.PAS aus der c't         *)
```

```
function UBIOS(fn,pa,pbc,pde,phl:integer):integer;
var biospb : record
               func,a    : byte;
               bc,de,hl  : integer;
               end;
    result : integer;
begin
  with biospb do begin
    func:=fn; a:=pa;
    bc:=pbc; de:=pde; hl:=phl;
    end;

      result:=BDOS(50,addr(biospb));

    ubios:=result;
  end;

(* Schaltet eine Grafikseite ein *)

Procedure Gra_On (seite : byte);
var  sys0,sys1   :    byte;
begin
  sys0 := portX2490;
   if (seite = 0) then   portX2490 := sys0 and 239
                  else   portX2490 := sys0 or   16;
  sys1 := portX2500;
  portX2500 := sys1 or 2;
end;

(* ------------------------------------------------------------- *)

(* Schaltet eine Grafikseite aus *)

Procedure Gra_Off (seite : byte);
var  nixwert   :    integer;
begin
nixwert:=ubios(30,0,(seite*256)+25,0,0);
end;
(* -------------------------------------------------------------*)

(* Fuellt die Grafikseite mit FARBE *)

procedure Gra_Cls(seite,farbe : byte);
var  nixwert   :    integer;
begin
nixwert := UBIOS(30,farbe,(seite*256)+26,0,0);
end;
(* ------------------------------------------------------------- *)

(* Setzt einen nicht normierten Punkt auf SEITE *)

Procedure Set_Point(xkoor,ykoor : integer; seite : byte);
var  nixwert   :    integer;
begin
nixwert := ubios(30,1,(seite*256)+27,xkoor,ykoor);
end;
(* ------------------------------------------------------------- *)
```



```
(* Loescht einen nicht normierten Punkt auf SEITE *)

Procedure Res_Point(xkoor,ykoor : integer; seite : byte);
var  nixwert   :    integer;
begin
nixwert := UBIOS(30,0,(seite*256)+27,xkoor,ykoor);
end;
(* ------------------------------------------------------------- *)


(* Gibt 1 zurueck wenn der nicht normierte Punkt gesetzt ist *)

Function Ask_Point(xkoor,ykoor : integer;seite : byte) :  integer;
begin
ask_point := UBIOS(30,0,(seite*256)+28,xkoor,ykoor);
end;
(* ------------------------------------------------------------- *)

(* Zieht eine Linie auf SEITE mit FARBE *)

Procedure Hrg_Line(xstart,ystart,xend,yend : integer; farbe,seite : byte

var  parablock : arrayX0..40 of integer;
     nixwert,z :                integer;

begin
parablockX10 := xstart;
parablockX20 := ystart;
parablockX30 := xend;
parablockX40 := yend;
z := addr(parablockX10);
nixwert := UBIOS(30,farbe,(seite*256)+29,0,z);
end;
(* ------------------------------------------------------------- *)

(* Setzt normierten Punkt auf SEITE *)

Procedure Set_Point_Normiert(xkoor,ykoor : integer; seite : byte);
var  nixwert   :    integer;
begin
nixwert := ubios(30,17,(seite*256)+27,xkoor,ykoor);
end;
(* ------------------------------------------------------------- *)

(* Loescht normierten Punkt auf SEITE *)

Procedure Res_Point_Normiert(xkoor,ykoor : integer; seite : byte);
var  nixwert   :    integer;
begin
nixwert := UBIOS(30,16,(seite*256)+27,xkoor,ykoor);
end;
(* ------------------------------------------------------------- *)

(* Zeichnet normierte Linie mit FARBE auf SEITE *)

Procedure Hrg_Line_Normiert(xstart,ystart,
                            xend,yend : integer; farbe,seite :byte);
var  parablock : arrayX0..40 of integer;
     nixwert,z :                integer;
begin
```

```
parablockA1Ü := xstart;
parablockA2Ü := ystart;
parablockA3Ü := xend;
parablockA4Ü := yend;
z := addr(parablockA1Ü);
nixwert := UBIOS(30,16+farbe,(seite*256)+29,0,z);
end;
(* ---------------------------------------------------------------- *)

(* Zeichnet einen Kreis auf SEITE mit FARBE *)

Procedure Hrg_Circle(xkoor,ykoor,radius : integer; farbe,seite : byte);
var parablock : arrayA1..3Ü of integer;
    nixwert,z :                integer;
begin
parablockA1Ü := xkoor;
parablockA2Ü := ykoor;
parablockA3Ü := radius;
z := addr(parablockA1Ü);
nixwert := UBIOS(30,16+farbe,(seite*256)+30,0,z);
end;
(* ---------------------------------------------------------------- *)
```

Mit diesem Include File schafft man sich die Voraussetzungen zur weiteren Programmierung. Nicht implementiert ist eine Funktion, die eigentlich beim Genie IIIs unter Holte CP/M nie genutzt wurde - die Window-Funktion. Window Technik unter CP/M ist ja möglich (es sei nur an die CHIP Turbo Pascal Hefte erinnert), aber Speicherintensiv und mühselig zu programmieren. Als Genie IIIs Besitzer muß man sich darum nicht kümmern, denn eine Window Funktion ist im BIOS implemtiert. Wie das unter Turbo Pascal aussieht zeigt der Include File WINDOWG.BIB.

```
ä ================================================================ ü
ä             WINDOWG.BIB                                          ü
ä Bibliotheks-Modul fuer Windows unter Holte CP/M+                  ü
ä zusaetzlich wird, wie bei WINDOW(7).BIB die Routine WINDOW.PAR    ü
ä benoetigt. Es wurde versucht weitgehend die Syntax von            ü
ä WINDOW.BIB einzuhalten.                                           ü
ä ================================================================ ü
ä Egbert Schroeer u. Volker Dose Januar bis Juni 1993               ü
ä ================================================================ ü

var     anystring           :       stringA255Ü;
        buffer              :       arrayA0..20000Ü of byte;
        TempScr             :       arrayA0..20000Ü of byte;
        i,j                 :       byte;
        a1,b1,a2,b2,
        WindowBreite,
        WindowHoehe         : integer;


(* Direkter BIOS-Aufruf ueber BDOS-Funktion 50 *)
(* wurde hier auch definiert wenn GRAFIK.INC   *)
(* nicht benötigt                              *)

function UBIOS(fn,pa,pbc,pde,phl:integer):integer;
var biospb : record
               func,a    : byte;
               bc,de,hl : integer;
               end;
    result : integer;
begin
  with biospb do begin
    func:=fn; a:=pa;
    bc:=pbc; de:=pde; hl:=phl;
    end;
  result:=0;
  case fn of
    2,3,7,13..15,17..19,24 : result:=BDOS(50,addr(biospb));
    9,16,20,22,25          : result:=BDOSHL(50,addr(biospb));
    else                     BDOS(50,addr(biospb));
    end;
  ubios:=result;
  end;

procedure Save_Screen(zeiger_buffer : integer);
var   nixwert : integer;
begin
nixwert := ubios(30,0,22,0,zeiger_buffer);
end;

procedure Restore_Screen(zeiger_buffer : integer);
var nixwert : integer;
begin
  nixwert := ubios(30,1,22,0,zeiger_buffer);
end;

procedure Reduzier_Window;

begin
  a1:=a1+1;
  b1:=b1-1;
  a2:=a2+1;
  b2:=b2-1;
end;

(*****************************************************************)
(*        OpenWindow - Oeffnen eines Fensters                    *)
(*****************************************************************)
ä Procedure  OpenWindow
  eroeffnet auf dem Bildschirm einen Bereich mit neu definierten
  Bildfenstergroessen. Alle Bildschirmausgaben nach Aufruf
  dieser Funktion beziehen sich nur noch auf die neuen Bildschirm-
  groessen.                                                      ü

procedure OpenWindow (Window_Number,top_line,bottom_line,
                      left_column,right_column : byte);

const   window_char  =      'F';   ä Die uebergebenen Parameter ü
        set_top      =      'I';   ä muessen mit 32 addiert     ü
        set_bottom   =      'J';   ä werden, steht so im HOLTE  ü
        set_left     =      'K';   ä Handbuch CPM3.DOC.         ü
        set_right    =      'L';   ä Durch die Definition der   ü
        escape       =      #27;   ä Konstanten innerhalb der   ü
                                   ä Prozedur kann sie einfach  ü
                                   ä 'so' mit $I eingebunden    ü
                                   ä werden                     ü
```

```
begin
  a1 := top_line;          ä zunaechst Daten sichern zur
                           Initialisierung des eigentlichen Fensters ü
  b1 := bottom_line;
  a2 := left_column;
  b2 := right_column;

  Cursor_Off;                    ä der stoert jetzt nur ü

  ä Breite und Hoehe des Fensters ermitteln          ü
  ä Davon 2 Zeichen fuer LinksOben, RechtsOben usw.  ü
  ä abziehen                                         ü

  WindowBreite := right_column - left_column-1;
  WindowHoehe  := bottom_line - top_line-1;

  ä Fenster fuer Rahmen initialisieren ü

  write(escape,window_char,chr(Window_Number+32));
  write(escape,set_top,chr(top_line+32));
  write(escape,set_bottom,chr(bottom_line+32));
  write(escape,set_left,chr(left_column+32));
  write(escape,set_right,chr(right_column+32));
  ClrScr;       ä Fenster sauber machen  ü

  ä Rahmen zeichnen          ü

  write(LinksOben);
  for i:=1 to WindowBreite do write(WaagrechtO);
  writeln(RechtsOben);
  for j := 1 to WindowHoehe do
    begin
      write(SenkrechtL);
      gotoxy(WindowBreite+2,j+1);
      writeln(SenkrechtR);
    end;
  write(LinksUnten);
  for i:=1 to WindowBreite do write(WaagrechtU);
  write(RechtsUnten);

ä Und nun die Fenstergroesse reduzieren um den
  Rahmen zu erhalten ü

  Reduzier_Window;
  write(escape,window_char,chr(Window_Number+32));
  write(escape,set_top,chr(a1+32));
  write(escape,set_bottom,chr(b1+32));
  write(escape,set_left,chr(a2+32));
  write(escape,set_right,chr(b2+32));
  ClrScr;
  Cursor_On;        ä den brauchen wir jetzt wieder ü
end;
```

```
(*****************************************************************)
(*     ExitWindows - stellt urspruenglichen Bildschirminhalt her *)
(*****************************************************************)

procedure ExitWindow;
const   window_char  =      'F';   ä Die uebergebenen Parameter ü
        set_top      =      'I';   ä muessen mit 32 addiert     ü
        set_bottom   =      'J';   ä werden, steht so im HOLTE  ü
        set_left     =      'K';   ä Handbuch CPM3.DOC.         ü
        set_right    =      'L';   ä Durch die Definition der   ü
        escape       =      #27;   ä Konstanten innerhalb der   ü
                                   ä Prozedur kann sie einfach  ü
                                   ä 'so' mit $I eingebunden    ü
                                   ä werden                     ü
begin
  write(escape,window_char,chr(0+32));
  write(escape,set_top,chr(0+32));
  write(escape,set_bottom,chr(24+32));
  write(escape,set_left,chr(0+32));
  write(escape,set_right,chr(80+32));
  clrscr; (* alles sauber hinterlassen *)
  Restore_Screen(addr(buffer)); (* urspr. Anzeige restaurieren *)
end;
```

Nun besitzen wir zwei Include Files, die es uns ermöglichen alle im Holte BIOS implementierten Grafik-Funktionen auszunutzen. Als erstes Beispiel das Programm GRAFIK.PAS, das einfach einige Aktionen mit der Grafik durchführt:

```
program grafik;

var   fn,pa,pbc,pde,phl  :          integer;
      xkoor,ykoor        :          integer;
      dummi              :          integer;
      sys0,sys1          :             byte;
      radius,i           :          integer;
      taste              :             char;
      hugo               :          boolean;

ä$I c5:grafik.incü

procedure raumschiff(xkoor,ykoor : integer; farbe,seite : byte);
begin
  hrg_line(xkoor   ,ykoor,   xkoor+29,ykoor,   farbe,seite);
  hrg_line(xkoor+4 ,ykoor+1,xkoor+25,ykoor+1,farbe,seite);
  hrg_line(xkoor+4 ,ykoor-1,xkoor+25,ykoor-1,farbe,seite);
  hrg_line(xkoor+13,ykoor-2,xkoor+16,ykoor-2,farbe,seite);
  hrg_line(xkoor+11,ykoor+2,xkoor+12,ykoor+2,farbe,seite);
  hrg_line(xkoor+17,ykoor+2,xkoor+18,ykoor+2,farbe,seite);
  hrg_line(xkoor+12,ykoor+3,xkoor+12,ykoor+3,farbe,seite);
  hrg_line(xkoor+17,ykoor+3,xkoor+17,ykoor+3,farbe,seite);
  hrg_line(xkoor+13,ykoor+4,xkoor+16,ykoor+4,farbe,seite);
end;



begin                    (* Hauptprogramm  *)
clrscr;

writeln('Testprogramm um die HRG einzuschalten ');

gotoxy(1,2);  clreol;
writeln('Jetzt wird die HRG eingeschaltet !');
  gra_cls(0,0);
  gra_cls(1,0);
```

```
  gra_on(0);
repeat until keypressed;


gotoxy(1,2);  clreol;
writeln('Jetzt werden Punkte gesetzt ! Und zwar normiert !');
  set_point_normiert(200,100,0);
  set_point_normiert(201,102,0);
  set_point_normiert(202,100,0);
  set_point_normiert(203,102,0);
  set_point_normiert(204,100,0);
  set_point_normiert(205,102,0);
  set_point_normiert(206,100,0);
  set_point_normiert(207,102,0);
  set_point_normiert(208,100,0);
  set_point_normiert(209,102,0);
repeat until keypressed;


gotoxy(1,2);  clreol;
writeln('Jetzt werden sie wieder gelöscht !');
  res_point_normiert(200,100,0);
  res_point_normiert(201,102,0);
  res_point_normiert(202,100,0);
  res_point_normiert(203,102,0);
  res_point_normiert(204,100,0);
  res_point_normiert(205,102,0);
  res_point_normiert(206,100,0);
  res_point_normiert(207,102,0);
  res_point_normiert(208,100,0);
  res_point_normiert(209,102,0);
repeat until keypressed;

gotoxy(1,2);  clreol;
writeln('Jetzt werden nicht normierte Punkte gesetzt.');
  set_point(638,174,0);
  set_point(639,174,0);
  set_point(638,175,0);
  set_point(639,175,0);

randomize;

for i := 0 to 500 do begin
                        xkoor := random(639);
                        ykoor := random(275);
                        set_point(xkoor,ykoor,0);
                      end;

repeat until keypressed;

gotoxy(1,2);  clreol;
write('Der normierte Punkt (639,174) ist ');

if ask_point(639,174,0)=1 then writeln('gesetzt.')
                          else writeln('nicht gesetzt.');

repeat until keypressed;
```



```
gotoxy(1,2);  clreol;
writeln('Und jetzt wird mit SetPoint eine Linie gezogen !');

for xkoor := 200 to 250 do
  for ykoor := 300 to 350 do set_point_normiert(xkoor,ykoor,0);

repeat until keypressed;

gotoxy(1,2);  clreol;
writeln('Und jetzt Linien mit HrgLine !!        Schneller !!');
  hrg_line_normiert(0,225,639,225,1,0);
  hrg_line_normiert(315,0,315,449,1,0);

  for xkoor:= 200 to 300 do hrg_line_normiert(xkoor,
                                              110,xkoor,210,1,0);

repeat until keypressed;


gotoxy(1,2);  clreol;
write   ('Jetzt noch einige Kreise !!');
writeln ('  Sie werden immer normiert gezeichnet.');
  hrg_circle(50,50,25,1,0);
  hrg_circle(100,50,25,1,0);
  hrg_circle(150,50,25,1,0);
  hrg_circle(50,100,25,1,0);
  hrg_circle(100,100,25,1,0);
  hrg_circle(150,100,25,1,0);
  hrg_circle(320,220,188,1,0);
repeat until keypressed;


gotoxy(1,2);  clreol;
writeln('Jetzt wird Seite 1 eingeschaltet !!');
  gra_cls(1,0);
  gra_on(1);
repeat until keypressed;

gotoxy(1,2);  clreol;
writeln('Wir loeschen den Schirm mit weiss !!');
  gra_cls(1,1);
repeat until keypressed;

gotoxy(1,2);   clreol;
writeln('Es folgen einige Grafikspielereien.');

randomize;

for i := 0 to 4000 do
    begin
      xkoor := 200+random(239);
      ykoor := 100+random(249);
      res_point_normiert(xkoor,ykoor,1);
    end;


for ykoor := 130 to 320 do hrg_line_normiert(230,ykoor,
                                             410,ykoor,1,1);
```

```
radius := 1;
while  radius < 60 do begin
                        radius := radius +3;
                        hrg_circle(160,225,radius,0,1);
                        hrg_circle(480,225,radius,0,1);
                        hrg_circle(320,225,radius,0,1);
                      end;

radius := 64;
while radius >1 do      begin
                        radius := radius  -3;
                        hrg_circle(160,225,radius,1,1);
                        hrg_circle(480,225,radius,1,1);
                        hrg_circle(320,225,radius,1,1);
                      end;


for i:= 1 to 100 do     begin
                        radius := 2 + random(20);
                        xkoor := 23 + random(600);
                        ykoor := 23 + random(410);
                        hrg_circle(xkoor,ykoor,radius,0,1);
                      end;

gra_cls(1,0);

randomize;

for i := 0 to 200 do begin  xkoor := random(639);
                            ykoor := random(274);
                            set_point(xkoor,ykoor,1);
                            end;


xkoor := 200;
ykoor := 100;

for i := 1 to 25 do begin xkoor := xkoor + 15;
                          ykoor := ykoor +  5;
                          raumschiff(xkoor,ykoor,1,1);
                          raumschiff(xkoor,ykoor,0,1);
                          end;

for i:= 1 to 70 do begin  ykoor := ykoor - 3;
                          raumschiff(xkoor,ykoor,1,1);
                          raumschiff(xkoor,ykoor,0,1);
                          end;

raumschiff(xkoor,ykoor,1,1);

gotoxy(1,2);  clreol;
writeln('Raumschiff bewegen mit Ziffernblock, ENDE mit <x>.');

hugo := true;

while ( hugo = true ) do begin
                           repeat until keypressed;
                           read(kbd,taste);
                           case taste of
                           '4' : begin
                                   raumschiff(xkoor,ykoor,0,1);
                                   xkoor:=xkoor-10;
                                   raumschiff(xkoor,ykoor,1,1);
                                 end;
                           '6' : begin
                                   raumschiff(xkoor,ykoor,0,1);
                                   xkoor := xkoor+10;
                                   raumschiff(xkoor,ykoor,1,1);
                                 end;
                           '8' : begin
                                   raumschiff(xkoor,ykoor,0,1);
                                   ykoor := ykoor +5;
                                   raumschiff(xkoor,ykoor,1,1);
                                 end;
                           '2' : begin
                                   raumschiff(xkoor,ykoor,0,1);
                                   ykoor := ykoor -5;
                                   raumschiff(xkoor,ykoor,1,1);
                                 end;
                           '7' : begin
                                   raumschiff(xkoor,ykoor,0,1);
                                   ykoor := ykoor + 5;
                                   xkoor := xkoor -10;
                                   raumschiff(xkoor,ykoor,1,1);
                                 end;
                           '1' : begin
                                   raumschiff(xkoor,ykoor,0,1);
                                   ykoor := ykoor - 5;
                                   xkoor := xkoor -10;
                                   raumschiff(xkoor,ykoor,1,1);
                                 end;
                           '9' : begin
                                   raumschiff(xkoor,ykoor,0,1);
                                   ykoor := ykoor + 5;
                                   xkoor := xkoor +10;
                                   raumschiff(xkoor,ykoor,1,1);
                                 end;
                           '3' : begin
                                   raumschiff(xkoor,ykoor,0,1);
                                   ykoor := ykoor - 5;
                                   xkoor := xkoor +10;
                                   raumschiff(xkoor,ykoor,1,1);
                                 end;
                           'x' : hugo := false;
                          end;
                         end;
gra_off(1);
gotoxy(1,2);
clreol;
writeln('Das wars dann.');
end.
```

Die mächtige Window-Funktion kann man beispielsweise im MINITED.PAS einsetzen, einem kleinen Texteditor. Durch den Einsatz der Window Technik bekommt er gleich ein sehr professionelles Aussehen. Auf alle anderen Include-Files wollen wir hier aus Platzgründen nicht eingehen. Es sei da auf die im Programmtext genannte Originalliteratur verwiesen.

```
ä ========================================================== ü
ä Diese Version von MiniTed ist nur fuer Genie 3s unter Holte   ü
ä CP/M 3.0. Herr Holte hat ueber Bios Funktion 30 den Zugriff   ü
ä auf Fenstertechnik eingebaut.                                 ü
ä Um Verbreitung unter Genie 3s Benutzern wird ausdruecklich    ü
ä gebeten.                                                      ü
ä Cursor Position muesste noch gespeichert werden, um nach      ü
ä restore_screen den Cursor an die "richtige" Position zu       ü
ä bringen.                                                      ü
ä Copyright: Die urspruengliche Version von MiniTed wurde in
             CHIP Turbo Pascal Special Ausgabe 9 veroeffentlicht.
             Von mir stammt nur die Erweiterung um die Fenster-
             technik. Ansprung des Bios und erste Gehversuche
             mit Holte CP/M und Window stammen von Volker Dose.
             Egbert Schroeer
             Joachimstrasse 18
             4270 Dorsten                                       ü

PROGRAM MiniTed;

(*$I c5:CPM-80.BIB    *)  ä                                     ü
(*$I c5:WINDOW.PAR    *)  ä hier Modifikationen fuer Zeichensatz ü
(*$I c5:WINDOWG.BIB   *)  ä Window fuer Genie 3s                ü
(*$I c5:WINDEFMI.INC  *)  ä Window Definitionen fuer TED        ü
(*$I c5:READCHAR.INC  *)  ä Angepasste Tastatureingabe          ü
(*$I c5:DYNSTR.BIB    *)  ä Dynamische Strings fuer Turbo-Pascal ü
(*$I c5:TED-1.INC     *)  ä Deklarationen von TED               ü
(*$I c5:TED-2.INC     *)  ä Der erste Teile von EditText        ü
(*$I c5:CTRLQ.INC     *)  ä Das Ctrl-Q-Menue                    ü
(*$I c5:CTRLK.INC     *)  ä Das Ctrl-K-Menue                    ü
(*$I c5:TED-3.INC     *)  ä Der zweite Teil von EditText        ü

procedure Logo;
  begin
      save_screen(addr(buffer));
      ClrScr; Write  (´     MINITED     fuer Genie 3s´);
              Writeln(´  Version 1.0 - ´#152´ ES (Juni 1993) ´);
  end;

(* Hauptproramm *)

VAR T : TextListe;
    f : Text;
    X, Y, Z : INTEGER;
    c : char;
    s : stringA080Ü;

BEGIN (* MiniTed *)
  CBreak:=FALSE;
  IF ParamCount=0 THEN
    BEGIN
      Logo;
      SelWindow(1);
      ClrScr;
      Write(´ Editfile : ´); readln(s);
      assign(f,s);
      SelWindow(2);gotoxy(1,1); (* Textbereich oeffnen *)
      ClrScr; Cursor_On;
    end
  else assign(f,ParamStr(1));
  (*$I-*) reset(f); (*$I+*)
  IF IOResult=0 THEN BEGIN close(f);
                           LiesText(f,T) end ELSE NeuerText(T);
  X:=1; Y:=1; Z:=1; ClrScr; EditText(T,X,Y,Z,79,25,TRUE);
  SelWindow(1);
  gotoxy(1,1); Write(´Speichern ? (J/N) ´); clreol;
  repeat read(kbd,c); c:=upcase(c) until c in [´J´,´N´Ü;
  write(c);
  if c=´J´ then SchreibText(f,T);
  LoescheText(T); ClrScr;
  ExitWindow;
  Cursor_On
end.  (* MiniTed *)
```


Im nächsten Teil werden wir ein Beispiel in der Sprache ANSI C vorstellen. Hier hat Alexander Schmidt - wie man an seinem PCX-Reader im letzten Info sehen konnte - einiges an Vorarbeit geleistet.

Das PCX-Format

Hier ist also nun die angedrohte Fortsetzung des Artikels über den PCX-Reader. Falls jemand eigene Versuche mit dem PCX-Format machen oder die Programm in andere Sprachen umsetzen will, möchte ich noch etwas näher auf die Codierung und auf den Header eingehen.

Die PCX-Codierung

Da es auch bei den DOSen Speicherplatz nicht zum Nulltarif gibt, speichert man die Bilder nicht 1:1 ab, sondern komprimiert sie. Die Codierung ist eine sog. Lauflängencodierung, oder Neudeutsch Run Length Coding, und funktioniert so, daß statt einer langen Folge identischer Bytes nur noch ein Zählbyte und ein Datenbyte geschrieben wird. Am besten arbeitet das Verfahren, wenn viele gleichartige Bytes hintereinander kommen, was bei Grafiken mit großen einfarbigen Flächen ja meist gegeben ist. Ein Beispiel:

45 45 45 45 45 45 45 45 07 00 00 00 00 00 00 00 00 00 00 00

Würde codiert ergeben

08 45 07 0B 00

Damit man nun Zähler und Datenbytes unterscheiden kann und nicht auch einzelnen Bytes ein Zählbyte spendieren muß, werden beim PCX-Format die Zähler mit gesetztem Bit 6 und 7 markiert, d.h. sie sind immer > C0h und man bekommt

C8 45 07 CB 00

Wer aufgepaßt hat merkt jetzt sicher, daß da was nicht stimmen kann, denn da die Daten beliebig sind, können auch ganz zufällig Bytes mit gesetztem Bit 6 und 7 auftauchen. Deutlicher wird das Problem bei der Bytefolge

FF FF FF FF FF FF FF F0 00 00 00 00 00 00

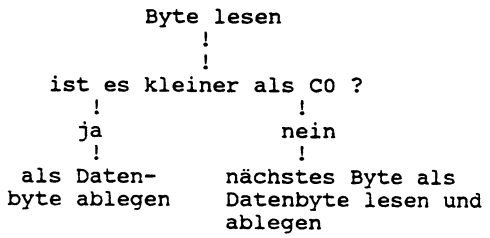
Nach obiger Anleitung codiert bekommt man

C7 FF F0 C6 00

Was nun ? Ist F0 jetzt ein Zähler, oder nicht ? Aber dann wäre C6 ein Datenbyte und alles käme durcheinander. In so einem Fall stellt man einfach doch noch ein Zählbyte voran und man erhält schließlich

C7 FF C1 F0 C6 00

Im Klartext bedeutet das also 7 mal FF, einmal F0 und 6 mal 00. Hier muß man also tatsächlich ein Byte zusätzlich spendieren, aber das braucht weniger Platz, als wenn man das grundsätzlich so machen und stur Zähler-Datenbyte schreiben würde. Beim Decodieren der Daten geht man nach folgendem Muster vor:



Wenn also ein Zählbyte erkannt ist, wird das nächste Byte als Datenbyte interpretiert, egal welche Bits gesetzt sind. Nochmal ein paar Bytes zum Üben:

0F C8 FF C1 E0 80 85 C9 00

Die Reihenfolge ist Datenbyte, Zähler, Datenbyte, Zähler, Datenbyte, Datenbyte, Datenbyte, Zähler, Datenbyte und die decodierte Folge lautet

0F FF FF FF FF FF FF FF FF E0 80 85 00 00 00 00 00 00 00 00 00

Der PCX-Header

Der Header im PCX-Format ist immer genau 128 Bytes lang und hat noch genug Freiraum, daß auch zukünftige Erweiterungen darin Platz haben. Die genaue Bedeutung der einzelnen Bytes ist anhand der Struktur im Programm ersichtlich, hier möchte ich nur noch ein paar ergänzende Anmerkungen machen.
Mit PCX kann man auch farbige Bilder codieren, wobei die Farbebenen wie die Folien auf einem Overheadprojektor übereinandergelegt werden. Dann müssen natürlich auch die Informationen für die Farbpalette belegt werden und hier liegt auch ein großer Nachteil des PCX-Formats. Da z.B. EGA und VGA-Karten völlig andere Paletten haben, bekommt man mit Sicherheit einen ganz anderen Farbeindruck wenn man Bilder zwischen zwei solcher Karten austauscht. Na ja, aber das kann uns wohl ziemlich egal sein.

Der PCX-Maker

Hier nun endlich das versprochene Programm, mit dem man auch selber Grafiken im PCX-Format abspeichern kann. Um den Bereich zu markieren, der abgespeichert werden soll, wird dieser mit einem Rechteck eingerahmt, das man über die Tastatur verschieben kann. Mit Kleinbuchstaben kann man die linke bzw. untere Kante verschieben, mit Großbuchstaben die rechte und die obere Kante. Die Bele-

gung ist wie beim WordStar üblich, E nach oben, X nach unten, S nach links und D nach rechts. R, C, A und F machen größere Sprünge in die entsprechende Richtung. An Grafikbefehlen braucht man die Abfrage eines Punktes und das Zeichnen eines Rechtecks, das man aber auch mit einem selbstgeschriebenen Unterprogramm machen kann. Als "Zeichenfarbe" nimmt man zweckmäßigerweise die zum Invertieren, sonst kommt der Bildschirminhalt ziemlich durcheinander.

```
/*****************************************************************/
/* PUTPCX.C                                                      */
/*                                                               */
/* Gets pixel data from the screen and writes a PCX format file */
/* Alexander Schmid 9/93                                         */
/*****************************************************************/

#include <stdio.h>
#include <conio.h>
#include <grafmod2.h>   /* computer specific graphic driver */

struct pcxheader{
      char manufact;      /*  0    Hersteller-ID           */
      char version;       /*  1    Versionsnummer          */
      char encode;        /*  2                            */
      char bpp;           /*  3                            */
      int xmin;           /*  4, 5 Bildgroesse             */
      int ymin;           /*  6, 7      "                  */
      int xmax;           /*  8, 9      "                  */
      int ymax;           /* 10,11      "                  */
      int hres;           /* 12,13 Aufloesung der Hardware */
      int vres;           /* 14,15     "       "      "    */
      char colormap[48];  /* 16-63                        */
      char reserved;      /* 64    momentan reserviert    */
      char ebene;         /* 65    Zahl der Farbebenen    */
      int bpline;         /* 66,67 Byte pro Zeile je Ebene */
      int paletinfo;      /* 68,69 Art des Bildes         */
      char blank[58];     /* derzeit keine Informationen  */
      } header;

/*---------------------------------------------------------------*/
/*                  write .PCX header into file                  */
/*---------------------------------------------------------------*/
writeheader(fp)
FILE *fp;
{
  int  count;
  char *loader;                 /* Zeiger auf ein character-array          */

  loader = (char*) &header;     /* LOADER zeigt auf den Header im Speicher */

  for(count = 0; count < 128; ++count){     /* 128 Bytes schreiben         */
    putc(*loader,fp);
    ++loader;                               /* Zeiger eins weiter          */
    }
}

/*---------------------------------------------------------------*/
/*                  read pixel from screen                       */
/*---------------------------------------------------------------*/
unsigned char readscreen(x,y)
int x,y;
{
  return point(x,YMAX-y);
}

/*---------------------------------------------------------------*/
/*                  mark area to write                           */
/*---------------------------------------------------------------*/
area(xl,yl,x2,y2)
int *xl,*yl,*x2,*y2;
{
  int xal,yal,xa2,ya2;
  char ch;

  hires();                                  /* Grafik einblenden    */
  color=2;                                  /* Punkt invertieren    */
  box(*xl,YMAX-*yl,*x2,YMAX-*y2);           /* Ausschnitt markieren */
  while((ch=getch())!=0x0D){                /* auf ENTER warten     */
    xal=*xl; yal=*yl; xa2=*x2; ya2=*y2;     /* Werte merken         */
    switch(ch){
      case 'd': if(*x2<XMAX) ++(*x2);       break;    /* rechter Rand ->  */
      case 'f': if(*x2<=XMAX-10) (*x2)+=10; break;    /* rechter Rand ->> */
      case 's': if(*x2>*xl+10) --(*x2);     break;    /* rechter Rand <-  */
      case 'a': if(*x2>*xl+20) (*x2)-=10;   break;    /* rechter Rand <<- */
      case 'x': if(*y2<YMAX) ++(*y2);       break;    /* unterer Rand V   */
      case 'c': if(*y2<YMAX-10) (*y2)+=10;  break;    /* unterer Rand VV  */
      case 'e': if(*y2>*yl+10) --(*y2);     break;    /* unterer Rand ^   */
      case 'r': if(*y2>*yl+20) (*y2)-=10;   break;    /* unterer Rand ^^  */

      case 'D': if(*xl<*x2-10) ++(*xl);     break;    /* linker Rand ->   */
      case 'F': if(*xl<*x2-20) (*xl)+=10;   break;    /* linker Rand ->>  */
      case 'S': if(*xl>0) --(*xl);          break;    /* linker Rand <-   */
      case 'A': if(*xl>=10) (*xl)-=10;      break;    /* linker Rand <<-  */
      case 'X': if(*yl<*y2-10) ++(*yl);     break;    /* oberer Rand V    */
      case 'C': if(*yl<*y2-20) (*yl)+=10;   break;    /* oberer Rand VV   */
      case 'E': if(*yl>0) --(*yl);          break;    /* oberer Rand ^    */
      case 'R': if(*yl>10) (*yl)-=10;       break;    /* oberer Rand ^^   */
      }
    box(xal,YMAX-yal,xa2,YMAX-ya2);  /* alten Rand loeschen        */
    box(*xl,YMAX-*yl,*x2,YMAX-*y2);  /* neuen Rand ziehen          */
    }
  box(*xl,YMAX-*yl,*x2,YMAX-*y2);    /* Markierung wieder loeschen */
  textmode();                        /* Grafik abschalten          */
  }

/*---------------------------------------------------------------*/
/*                  do run length encoding                       */
/*---------------------------------------------------------------*/
compress(arrl,bytes,fp)
unsigned char arrl[];
int bytes;
FILE *fp;
{
  unsigned char arr2[150],ch;
  int repeat,ausgabe=0,eingabe=0;

  while(eingabe<bytes){               /* Zahl der Eingangsbytes          */
    ch=arrl[eingabe];                 /* Byte aus dem Eingangsarray holen */
    ++eingabe;                        /* Eingabezeiger eins weiter       */
    repeat=1;                         /* Wiederholungszaehler            */
```

```
    while((ch==arr1[eingabe]) && (repeat<0x3F)){  /* Byte wiederholt sich */
      ++repeat; ++eingabe;                        /* Zaehler erhoehen     */
      }
    if(repeat>1){                          /* wenn es Wiederholungen gab   */
      arr2[ausgabe]=(char)repeat + 0xC0;   /* Zaehlbyte zusammensetzen     */
      ++ausgabe;                           /* Ausgabezaehler erhoehen      */
      }
    else{
      if((ch & 0xC0) == 0xC0){             /* Datenbyte mit gesetztem Bit 6 und 7 */
        arr2[ausgabe]=0xC1;                /* Zaehlbyte = 1 schreiben      */
        ++ausgabe;                         /* Ausgabezaehler erhoehen      */
        }
      }
    arr2[ausgabe]=ch;                      /* Byte in Ausgabearray schreiben */
    ++ausgabe;                             /* Ausgabezaehler erhoehen      */
    }
  for(eingabe=0; eingabe<ausgabe; ++eingabe)  /* Ausgabearray auf Floppy  */
    fputc(arr2[eingabe],fp);                  /* schreiben                */
  }

/*-------------------------------------------------------------------------*/
/*                    write .PCX data                                      */
/*-------------------------------------------------------------------------*/
writepcx(x1,y1,x2,y2,fp)
int x1,y1,x2,y2;
FILE *fp;
{
  int x,y,bits;
  unsigned char array[150],byte,count;

  hires();                                 /* Grafik einblenden           */
  for(y=y1; y<=y2; ++y){                   /* vom oberen zum unteren Rand */
    count=0;                               /* Zaehler fuer den Zeilenpuffer */
    for(x=x1; x<=x2; x+=8){                /* vom linken zum rechten Rand */
      byte=0;
      for(bits=0; bits<8; ++bits){         /* Byte zusammenbasteln        */
        byte |= readscreen(x+bits,y)<<7-bits;
        }
      array[count]=byte;
      ++count;
      }
    compress(array,(x2-x1+1)/8,fp);        /* Zeile komprimieren und auf  */
    }                                      /* Floppy schreiben            */
  textmode();                              /* Grafik abschalten           */
  }

/*------------------------- MAIN() ----------------------------------------*/

main( argc, argv )
int argc;
char *argv[2];
{
  FILE *fp;
  int x1,y1,x2,y2,xoff=0,yoff=0;
  char ch;

  argc = argc;

  if((fp = fopen(argv[1],"rb")) != NULL){
    printf("ERROR: FILE '%s' DOES ALREADY EXIST.\n",argv[1]);
    exit(0);
    }

  if((fp = fopen(argv[1],"wb")) == NULL){
    printf("ERROR: CAN NOT OPEN FILE '%s' \n",argv[1]);
    exit(0);
    }

  x1=0; y1=0; x2=200; y2=200;
  do{
    area(&x1,&y1,&x2,&y2);
    printf("Die Koordinaten sind %d/%d %d/%d\n",x1,y1,x2,y2);
    }while((ch=getch())==0x0D);   /* Korrektur mit RETURN moeglich */

  printf("\nX-/Y-Offsets so lassen ? ");
  if(((ch=getch() & 0xDF) == 'J') || (ch=='Y') || (ch==0x0D)){
    header.xmin=x1;
    header.ymin=y1;
    header.xmax=x2;
    header.ymax=y2;
    }
  else{
    printf("\n\n  X-Offset: "); scanf("%d",&xoff);
    printf("  Y-Offset: "); scanf("%d",&yoff);
    header.xmin=xoff;
    header.ymin=yoff;
    header.xmax=x2-x1+xoff;
    header.ymax=y2-y1+yoff;
    }

  header.encode=1;
  header.bpp=1;
  header.hres=XMAX;
  header.vres=YMAX;
  header.ebene=1;
  header.bpline=(x2-x1+1)/8;

  writeheader(fp);
  writepcx(x1,y1,x2,y2,fp);
  fclose(fp);

  exit(0);
  }
/* End of main() */
```

Fehler im PCXSHOW

Leider bemerkt man sowas immer erst, wenn man den Artikel schon abgeschickt hat, aber es ist nur eine Zahl. In der Routine READ-HEADER steht in der FOR-Schleife <127, da muß ein <128 hin. Ansonsten habe ich das Programm noch so geändert, daß man selber festlegen kann, wohin das Bild geladen werden soll. So kann man aus mehreren kleinen Bildern ein großes zusammensetzen. In der Discovery findet sich natürlich die neueste Version.

Alexander Schmid

Universelle Textersetzung

Es hat ja schon viele Programme gegeben, um Textdateien zwischen MSDOSen und CP/M-Rechnern auszutauschen, und besonders WordStar-Texte sind bei DOSen-Besitzern immer wieder gerne gesehen :-) Nachteilig bei solchen Programmen ist aber, daß ihr Funktionsumfang vom Programmierer genau festgelegt ist und sie nur ganz bestimmte, meistens einzelne, Zeichen umcodieren können. So hat man zwangsläufig bald eine größere Sammlung solcher Utilities, über die man schnell den Überblick verliert. Viel universeller ist nun ein Programm, das durch eine Parameterdatei gesteuert wird und so mehr oder weniger beliebige Strings gegen beliebige andere Strings austauschen kann. Hartmut Obermann und ich hatten so ein Programm vor Jahren mal von MSDOS nach CP/M umgesetzt und er hat mich jetzt gebeten, das auch für die vielen neuen Mitglieder (nochmal ?) im Info vorzustellen.

Das Parameterfile ist ein ganz gewöhnliches Textfile mit folgendem Aufbau:

```
--- dies ist ein Kommentar
alter String
neuer String
--- dies ist der nächste Kommentar
dritter String
vierter String
```

Die Stringlänge ist auf 80 Zeichen begrenzt, aber wenn das wirklich nicht reicht, muß man halt auf zweimal ersetzen. Steuerzeichen werden mit einem vorangestellten '^' eingegeben, also z.B. als ^I und um auch die IBM-Sonderzeichen ohne große Verrenkungen zu erreichen, wird bei einem vorangestellten '_' 64 zum Zeichencode des folgenden Zeichens addiert, d.h. das IBM-ä wird durch _D repräsentiert. Damit die IBM-Umlaute und sonstige Sonderzeichen mit Codes über 128 nicht von vornherein kaputtgemacht werden, maskiert GLOBAL das 8. Bit nicht aus, das muß nach wie vor mit einem getrennten Programm erfolgen. Wenn man also einen WordStar-Text nach MSDOS umsetzen will, muß man erst das 8. Bit mit einem anderen Programm löschen und dann GLOBAL aufrufen.

Natürlich gibt es für den Sonderfall DOS <-> CP/M spezielle Programme, die das Maskieren und Umcodieren viel einfacher und in einem Rutsch machen, aber das war hier nur als Beispiel gedacht und es gibt ja auch unter CP/M noch zahlreiche Textverarbeitungen, die teilweise recht seltsame Formate haben und für die es keine speziellen Umsetzungsprogramme gibt, damit man die Texte mit "normalen" Textprogrammen weiterverarbeiten kann.

Eine andere mögliche Anwendung mit durchaus praktischer Bedeutung könnte die Ersetzung von (persönlichen) Kürzeln gegen Text, z.B. "mfg" gegen "Mit freundlichen Grüßen" oder "sgh" gegen "Sehr geehrte Herren" sein. Wenn man solche Phrasen nicht auf Funktionstasten legen kann, kann man sich so einiges an Tipparbeit sparen. Wenn man will, kann man so sogar ganze Briefköpfe erzeugen, indem man z.B. sowas schreibt:

```
--- Brief an Anton
AntonAdr
Herrn^M^JAnton Meier^M^JKleine Str. 12^M^J12345 Posemuckel^M^J
```


Rekursionen sollte man dabei tunlichst vermeiden (darum steht oben AntonAdr und nicht Anton), sonst erhält man zwar sehr interessante, aber sicher nicht die gewünschten Ergebnisse.

```
--- vorsicht !
Anton
Hallo Anton, wie geht's ?
```

Das mit den IBM-Umlauten habe ich erst nachträglich eingefügt und in der Discovery liegt noch eine ältere Version, aber die Änderung in der Prozedur "InterpretCC" ist minimal und kann sehr einfach durch Kopieren des vorherigen Blocks und etwas editieren gemacht werden. Wenn wir schon bei den Änderungen sind, wie ich gerade gemerkt habe, hat die Version in der Discovery auch noch einen Bug, der verhindert, daß das weiter oben mit dem Briefkopf richtig funktioniert. Der Bug ist auch in "InterpretCC", sodaß Ihr die Patches sehr leicht machen könnt, wenn Ihr Euch das File aus der Discovery holt.

Der Aufruf des Programms erfolgt mit

"GLOBAL eingabefile ausgabefile parameterdatei"

oder

"GLOBAL eingabefile ausgabefile parameterdatei I"

wobei im zweiten Fall noch die IBM-Umlaute in ASCII-Umlaute umgewandelt werden. Im Prinzip ist das überflüssig, weil man es mit einem Parameterfile genausogut machen kann, aber es ist nunmal im Programm drin. Das Ausgabefile darf noch nicht existieren, sonst wird das Programm mit einer Fehlermeldung abgebrochen.

```
PROGRAM Global_Replace;

(* mit Umsetzung der IBM-Umlaute in ASCII-Umlaute *)

(*$I bytefile.bib *)

CONST Version='Version 1.1.2';
      InputLength=80;

TYPE String80=STRING[80];
     String255=STRING[255];
     T_WorkFiles=ByteFile;   (* statt FILE OF BYTE *)
     T_DefsFile=TEXT;
     P_ReplDefs=^T_ReplDefs;
     T_ReplDefs=RECORD
                  OldString:String80;
                  NewString:String80;
```

```
                Next     :P_ReplDefs;
              END;

VAR InputLine,OutputLine:String255;
    InputFile,OutputFile:T_WorkFiles;
    ReplDefsFile        :T_DefsFile;
    ReplDefs            :P_ReplDefs;
    MaxReplString,i     :BYTE;
    IBM                 :BOOLEAN;

PROCEDURE Message;
BEGIN
  WRITELN('GLOBAL REPLACE (',Version,')');
  WRITELN;
  WRITELN('(c) Hans Wiederhold 1986');
  WRITELN
END;

PROCEDURE Oops;
BEGIN
  WRITELN('Aufruf mit "GLOBAL InputFile OutputFile ReplaceDefs [I]" ');
  WRITELN('Ein File mit dem Namen der Output-Datei darf nicht existieren.');
  WRITELN;
  WRITELN('Das ReplaceDefs-File hat folgendes Format:');
  WRITELN;
  WRITELN('--- Kommentar fuer den Benutzer, wird nicht interpretiert');
  WRITELN('zu ersetzender String');
  WRITELN('einzufuegender String');
  WRITELN('--- naechster Kommentar usw.');
  WRITELN;
  WRITELN('Control-Zeichen werden mit ^-Zeichen und ^ mit ^! eingegeben');
  WRITELN;
  WRITELN('Mit dem Parameter I werden IBM-Umlaute in ASCII-Umlaute umcodiert');
  WRITELN;
  HALT
END;

PROCEDURE FilesInit(VAR Input:T_WorkFiles; VAR Output:T_WorkFiles;
                    VAR ReplaceDefsFile:T_DefsFile);
VAR InputFileName,OutputFileName,DefsFileName:String80;
  PROCEDURE UpperCase(VAR WorkString:String80);
  VAR i:BYTE;
  BEGIN
    FOR i:=1 TO length(WorkString) DO
      WorkString[i]:=UpCase(WorkString[i])
  END;
BEGIN
  IF ParamCount<3 THEN Oops;
  IBM:=(ParamStr(4)='I');
  InputFileName:=ParamStr(1);
  (*V-*) UpperCase(InputFileName); (*V+*)
  OutputFileName:=ParamStr(2);
  (*V-*) UpperCase(OutputFileName); (*V+*)
  DefsFileName:=ParamStr(3);
  (*V-*) UpperCase(DefsFileName); (*V+*)

  ASSIGN(ReplaceDefsFile,DefsFileName);
  (*$I-*) RESET(ReplaceDefsFile); (*$I+*)
  IF (IOresult<>0) THEN Oops;
  AssignByteFile(Input,InputFileName);
  (*$I-*) ResetByteFile(Input); (*$I+*)
  IF (IOresultByteFile<>0) THEN Oops;
  AssignByteFile(Output,OutputFileName);
  (*$I-*) ResetByteFile(Output); (*$I+*)
  IF (IOresultByteFile=0) THEN Oops;
  RewriteByteFile(Output)
END;

PROCEDURE LoadReplacements(VAR ReplaceDefsFile:T_DefsFile;
                           VAR FirstDef:P_ReplDefs; VAR MaxIstLength:BYTE);
VAR LastDef,NewDef:P_ReplDefs;
    Dummy,Ist,Soll:String80;

  PROCEDURE InterpretCC(VAR WorkString:String80);
  VAR CCPos,CCh,i:BYTE;
  BEGIN
    i:=1;
    REPEAT
      CCPos:=POS('^',COPY(WorkString,i,LENGTH(WorkString)-i+1));
      IF CCPos>0 THEN
      BEGIN
        CCh:=ORD(WorkString[CCPos+i]);
        IF CCh=33 THEN CCh:=94 ELSE CCh:=(CCh AND 159);
        WorkString[CCPos+i-1]:=CHR(CCh);
        IF CCh=95 THEN i:=i+CCPos+1 ELSE i:=i+CCPos;
        DELETE(WorkString,i,1);
      END;
    UNTIL CCPos=0;
    REPEAT
      CCPos:=POS('_',COPY(WorkString,i,LENGTH(WorkString)-i+1));
      IF CCPos>0 THEN
      BEGIN
        CCh:=ORD(WorkString[CCPos+i]);
        IF CCh=33 THEN CCh:=95 ELSE CCh:=(CCh + 64);
        WorkString[CCPos+i-1]:=CHR(CCh);
        IF CCh=95 THEN i:=i+CCPos+1 ELSE i:=i+CCPos;
        DELETE(WorkString,i,1);
      END;
    UNTIL CCPos=0;
  END;

BEGIN
  Ist:=''; Soll:='';
  MaxIstLength:=0;
  FirstDef:=NIL;
  WHILE NOT EOF(ReplaceDefsFile) DO
  BEGIN
    READLN(ReplaceDefsFile,Dummy);  (* Kommentar *)
    IF (NOT EOF(ReplaceDefsFile)) THEN READLN(ReplaceDefsFile,Ist);
    IF (NOT EOF(ReplaceDefsFile)) THEN READLN(ReplaceDefsFile,Soll);
    IF ((Ist<>'') AND (Ist<>Soll)) THEN
    BEGIN
      interpretCC(Ist);
      interpretCC(Soll);
      NEW(NewDef);
      NewDef^.OldString:=Ist;
      NewDef^.NewString:=Soll;
      IF (FirstDef=NIL) THEN FirstDef:=NewDef ELSE LastDef^.Next:=NewDef;
      LastDef:=NewDef;
      LastDef^.Next:=NIL
```

```
    END;
    IF LENGTH(Ist)>MaxIstLength THEN MaxIstLength:=LENGTH(Ist)
  END
END;

PROCEDURE GetInputLine(VAR Input:String255; VAR InputFile:T_WorkFiles);
CONST InputLength=80;
VAR Ch:BYTE;
BEGIN
  WHILE ((LENGTH(Input)<InputLength) AND (NOT EofByteFile(InputFile))) DO
  BEGIN
    ReadByteFile(InputFile,Ch);
    IF IBM THEN
      Case Ch OF
        $15 : Ch:=$40; (* '@' *)
        $8E : Ch:=$5B; (* '[' *)
        $99 : Ch:=$5C; (* '\' *)
        $9A : Ch:=$5D; (* ']' *)
        $84 : Ch:=$7B; (* '{' *)
        $94 : Ch:=$7C; (* '|' *)
        $81 : Ch:=$7D; (* '}' *)
        $E1 : Ch:=$7E; (* '~' *)
    END;
    Input:=Concat(Input,CHR(Ch))
  END
END;

PROCEDURE PutOutputLine(VAR Output:String255; VAR OutputFile:T_WorkFiles);
VAR i,Ch:BYTE;
BEGIN
  FOR i:=1 TO LENGTH(Output) DO
  BEGIN
    Ch:=ORD(Output[i]);
    WriteByteFile(OutputFile,Ch)
  END
END;

PROCEDURE DoReplacements(VAR Input:String255; VAR Output:String255;
                         Replacements:P_ReplDefs);
VAR PattPos:BYTE;
BEGIN
  Output:=Input;
  WHILE Replacements<>NIL DO
  BEGIN
    REPEAT
      PattPos:=Pos(Replacements^.OldString,Output);
      IF ((PattPos>0) AND (PattPos<=(LENGTH(Output)-MaxReplString))) THEN
      BEGIN
        DELETE(Output,PattPos,LENGTH(Replacements^.OldString));
        INSERT(Replacements^.NewString,Output,PattPos)
      END;
    UNTIL ((PattPos=0) OR (PattPos>=(LENGTH(Output)-MaxReplString)));
    Replacements:=Replacements^.Next
  END;
  Input:=COPY(Output,LENGTH(Output)-MaxReplString+1,MaxReplString);
  IF NOT EofByteFile(InputFile) THEN
    DELETE(Output,LENGTH(Output)-MaxReplString+1,MaxReplString);
END;
```



```
PROCEDURE FilesClose(VAR Input:T_WorkFiles; VAR Output:T_WorkFiles;
                     VAR ReplaceDefs:T_DefsFile);
BEGIN
  CLOSE(ReplaceDefs);
  CloseByteFile(Input);
  CloseByteFile(Output);
END;

BEGIN
  Message;
  FilesInit(InputFile,OutputFile,ReplDefsFile);
  LoadReplacements(ReplDefsFile,ReplDefs,MaxReplString);
  InputLine:=''; OutputLine:='';
  WHILE NOT EofByteFile(InputFile) DO
  BEGIN
    GetInputLine(InputLine,InputFile);
    DoReplacements(InputLine,OutputLine,ReplDefs);
    PutOutputLine(OutputLine,OutputFile)
  END;
  FilesClose(InputFile,OutputFile,ReplDefsFile)
END.
```

Wer das Listing nicht abtippen will, kann es wie immer aus der Discovery downloaden (GLOBAL.LBR) oder von mir oder Hartmut bekommen.

Alexander Schmid

Oberflächenmenü

Menüdatei unter DOS selbst erstellen

Viele Leser kennen Menüoberflächen, mit denen u.a. einfach und bequem Dateien aufgerufen werden können. Ein speicherplatzsparendes und effektives Menü bieten Ihnen die nachstehenden Dateien.
Zur Erstellung des Oberflächenmenüs gehen Sie wie folgt vor:

1. Schreiben Sie - am besten über einen Editor oder ein Textprogramm - nachstehende Zeilen ab und speichern diese als ASCII-Text unter dem Namen "WEITER.DAT". Die nach INT 21 stehende Leerzeile muß als Leerzeile erfaßt werden.

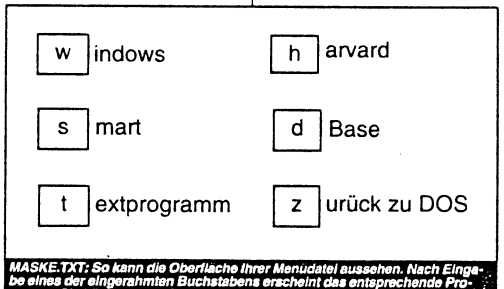
```
A
MOV AH,07
INT 21
CMP AL,00
JNZ 010A
JMP 0100
MOV AH,4C
INT 21

R CX
10
N WEITER.COM
W
Q
```

2. Starten Sie den DOS-Debugger (ein im DOS-Betriebssystem enthaltenes Programm) mit "debug < weiter.dat" (ohne Anführungszeichen!) gefolgt von einem return. Bei fehlerfreier Arbeit sollte die Datei "WEITER.COM" erzeugt sein. Wenn WEITER.COM später fehlerfrei arbeitet, kann WEITER.DAT gelöscht werden.

3. Schreiben Sie - am besten über einen Editor oder ein Textprogramm - nachstehende Zeilen ab und speichern diese als ASCII-Text unter dem Namen "MENUE.BAT". Die in Großbuchstaben und kursiv geschriebenen Texte können Sie Ihren Verzeichnissen und Programmen entsprechend anpassen.
Die beim "errorlevel" genannten Zahlen entsprechen den Kleinbuchstaben laut ASCII-Tabelle. Diese wiederum korrespondieren in unserem kleinen Beispiel mit den im Textmenue "MASKE.TXT" eingerahmten Buchstaben. Wenn Sie später also "w" für WINDOWS aufrufen, entspricht dies dem errorlevel 119 = w. Durch die dann folgende Anweisung "... goto WINDOWS" wird in den Bereich ":WINDOWS" verzweigt. Dort wird dann die Datei "WIN" aufgerufen. Nach Ende von "WINDOWS" wird wieder zu "haupt" an den Antang verwiesen und Sie können das nächste Programm aufrufen.
Die Verweise nach haupt1 (z.B. "if errorlevel 121 goto haupt1") wurden erstellt, um irrtümliche Tasteneingaben abzufangen. Wenn also anstelle von "w" ein "q" gedrückt wird, bricht das Programm nicht ab sondern verzweigt an den Anfang und wartet auf eine "richtige" Anweisung. Sie können die abzufangenden Tasteneingaben selbstverständlich erweitern ("if errorlevel .... goto haupt1").

```
@echo off
:haupt
c:
cd\
cls
type c:\menue\maske.txt
:haupt1
c:\menue\weiter.com
:abfrage1
if errorlevel 122 goto DOS
if errorlevel 121 goto haupt1
if errorlevel 120 goto haupt1
if errorlevel 119 goto WINDOWS
if errorlevel 118 goto haupt1
if errorlevel 117 goto haupt1
if errorlevel 116 goto WORD
if errorlevel 115 goto SMART
if errorlevel 114 goto haupt1
if errorlevel 113 goto haupt1
rem weitere errorlevel möglich
if errorlevel 105 goto haupt1
if errorlevel 104 goto HARVARD
if errorlevel 103 goto haupt1
if errorlevel 102 goto haupt1
if errorlevel 101 goto haupt1
if errorlevel 100 goto DBASE
if errorlevel 99 goto haupt1
if errorlevel 98 goto haupt
if errorlevel 97 goto haupt
goto haupt1
:DBASE
c:
cd\DBASE
DBASE.EXE
goto haupt
:WORD
c:
cd\WORD
WORD.EXE
goto haupt
:WINDOWS
c:
cd\WINDOWS
WIN
goto haupt
:HARVARD
c:
cd\HARVARD
HARVARD.EXE
goto haupt
:SMART
c:
cd\SMART
SMART D
goto haupt
:DOS
cd\
cls
```



MASKE.TXT: So kann die Oberfläche Ihrer Menüdatei aussehen. Nach Eingabe eines der eingerahmten Buchstabens erscheint das entsprechende Programm. Sie können diese Textdatei natürlich auch ganz anders gestalten.

4. Schreiben Sie - am besten über einen Editor oder ein Textprogramm - u.a. Grafikkasten ab und speichern ihn als ASCII-Text unter dem Namen "MASKE.TXT".

5. Sie haben nun folgende Dateien geschrieben/erzeugt: **weiter.dat, weiter.com, menue.bat, maske.txt**; legen Sie diese vier Dateien in ein Unterverzeichnis C:\MENUE ("md MENUE").

6. Nehmen Sie sich nun Ihre Datei autoexec.bat vor und fügen dem PATH-Eintrag das Unterverzeichnis C:\MENUE an. (PATH=...;C:\MENUE;). Geben Sie schließlich in Ihrer autoexec.bat als letzte Zeile ein: C:\MENUE\MENUE.BAT. Fertig.

7. Starten Sie Ihren Rechner neu. Bei fehlerfreier Eingabe muß die unter Ziffer 4. beschriebene Maske erscheinen. Sie können nun Ihre gewünschte Datei aufrufen, indem Sie den entsprechenden Buchstaben eingeben. Nach Programmende geht es automatisch wieder in die Maske. Über "Z" wie "zurück zu DOS" gelangen Sie - wenn gewünscht - wieder in Ihr Hauptverzeichnis.

8. Passen Sie die Maske Ihren Bedürfnissen entsprechend an. Sie können über diese Maske natürlich auch Kopier- und Formatierbefehle automatisieren.

(erstellt nach Hinweis von Helmut Bruditz, MatAmtBw)

Compilerbau — Methoden und Werkzeuge

Gerald Schröder, e-mail: gschroed@dbis1.informatik.uni-hamburg.de

Oktober 1993

Der folgende Artikel ist ein (gekürzter) Teil eines Kapitels meiner Diplomarbeit über Syntaxerweiterungen in Programmiersprachen, die ich im August dieses Jahres fertiggestellt habe. Der Artikel knüpft an den Compilerbau-Artikel aus Info 41, S. 37+38, an, der die grundsätzliche Arbeitsweise eines Compilers beschreibt. Ich kenne zwar die dazugehörige Artikelserie nicht, vermute aber, daß dort beschrieben wird, wie ein Compiler *von Hand* gebaut wird, sprich: jede Funktion des Compilers wird in Pascal (oder einer anderen Programmiersprache) ausprogrammiert.

Dieser Artikel soll dagegen eine andere Möglichkeit zeigen: Ein Compiler kann (ganz oder teilweise) mit speziellen Sprachen *beschrieben* werden. Wenn so eine Beschreibung vorliegt, können *Werkzeuge* bzw. *Generatoren* (also bestimmte Programme) automatisch aus diesen Beschreibungen einen Compiler erzeugen (bzw. generieren). Vorteil dieser Methode ist, daß die Beschreibungen und Werkzeuge für die verschiedenen Teile eines Compilers angepaßt sind und eine Fehlerprüfung erlauben, während bei von Hand geschriebenen Compilern beliebige Fehler eingebaut werden können.

Nachteile gibt es natürlich auch: Wer kennt/lernt schon diese speziellen Sprachen? (Eine Programmiersprache kennt dagegen jeder.) Wer hat die Werkzeuge mal eben zur Hand? (Einen Compiler hat dagegen jeder.) Wer möchte schon auf die Tricks und die daraus resultierende Effizienz verzichten, die handgeschriebener Code gegenüber automatisch erzeugtem Code bietet?

Aus meiner Erfahrung kann ich darauf antworten: Ein handgeschriebener Compiler, der auch noch trickreich programmiert ist, strotzt vor Fehlern und ist nicht wartbar. Ich habe das bei einem Modula-2 Compiler von ca. 30.000 Zeilen mitgemacht und war kurz vor dem Mordanschlag (auf meine Vorgänger). Natürlich sehe ich ein, daß nicht jeder die Werkzeuge zur Hand hat (ich habe sie zu Hause auch nicht), aber das sollte uns nicht hindern, wenigstens die Methodik anzuwenden: Ein Compiler, der erst (formal) beschrieben und dann anhand dieser Beschreibungen programmiert wird, ist immer noch wartbarer als ein „gehackter" Compiler.

Last, but not least, sind solche Methoden und Werkzeuge auch nicht nur beim Compilerbau zu gebrauchen: Wer zum Beispiel ein Konvertierungsprogramm (Daten von einem Format in ein anderes übersetzen, also auch eine Art Compiler) geschrieben hat, weiß solche Beschreibungen auch zu schätzen. Natürlich gibt es weitere Anwendungen, aber das überlasse ich Eurer Phantasie.

Ach ja: Wer eine Programmieraufgabe sucht, kann sich ja mal daran versuchen, eines der beschriebenen Werzeuge zu implementieren. Hinweise dazu geben z. B. [Aho 88] oder [Waite 85]. Ich habe selber einen Scanner- und einen Parser-Generator geschrieben, und wenn ich es kann, kann es jeder. Als einführendes Werk in den Bau von *handgeschriebenen* Compilern kann ich [Wirth 86] empfehlen.

1 Compilermodell

Dieser Abschnitt erläutert ein Compilermodell, das den Aufbau eines Compilers beschreibt. Es handelt sich um ein klassisches Phasenmodell nach [Aho 88].

1.1 Grobstruktur eines Compilers

Ein Compiler hat die Aufgabe, einen Quelltext (*source text*) in einen maschinenabhängigen Zielkode (*target code*) zu übersetzen. Diese Aufgabe läßt sich in verschiedene Teilaufgaben zerlegen, die hintereinander ausgeführt werden können. Jede Teilaufgabe transformiert eine Repräsentation des Quelltextes in eine andere, wobei die letzte Repräsentation der Zielkode ist. Die Abarbeitung einer Teilaufgabe wird als Phase bezeichnet.

Dieses Modell beschreibt den *logischen* Aufbau eines Compilers. In einer Implementation können aus Effizienzgründen durchaus mehrere Phasen in einem Lauf (*pass*) zusammengefaßt und verschiedene Repräsentationen in einer Datenstruktur dargestellt werden.

Abbildung 1 zeigt die aus zwei Phasen und drei Repräsentationen bestehende Grobstruktur eines Compilers.

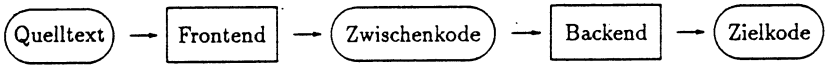


Abbildung 1: Grobstruktur eines Compilers

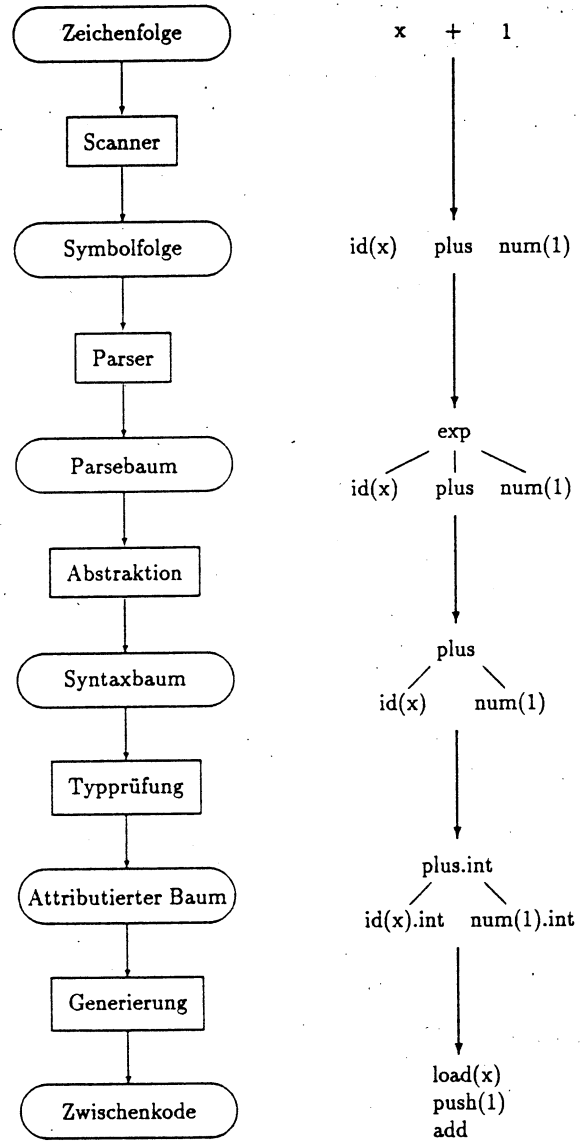
1. Das Frontend analysiert den Quelltext und transformiert ihn in einen maschinenunabhängigen Zwischenkode.
2. Das Backend synthetisiert aus dem Zwischenkode einen maschinenabhängigen Zielkode.

Ich betrachte im folgenden nur das Frontend, weil mich Backends einfach nicht interessieren (bzw. in meiner Diplomarbeit nicht das Thema waren).

1.2 Feinstruktur des Frontends eines Compilers

Das Frontend teilt sich in verschiedene Phasen auf, die im folgenden als Funktionen definiert werden. Abbildung 2 a) zeigt die Phasen und Repräsentationen des Frontends. Abbildung 2 b) enthält ein Beispiel für die Transformation der Datenstrukturen.

scan : *Characters* → *Symbols*
: Die lexikalische Analyse (Scanner) zerlegt den Quelltext, der als Zeichenfolge betrachtet wird, in eine Symbolfolge.



a) Phasen und Repräsentationen b) Beispiel

Abbildung 2: Feinstruktur eines Compiler-Frontends

parse : $Symbols \rightarrow ParseTree$
Die syntaktische Analyse (Parser) analysiert die Symbolfolge auf syntaktische Konstrukte, die als Parsebaum repräsentiert werden.

abstract : $ParseTree \rightarrow AST, \quad AST = AbstractSyntaxTree$
Durch die Abstraktion wird der Parsebaum in einen abstrakten Syntaxbaum transformiert.

check : $AST \rightarrow AST'$
Die statische semantische Analyse (Typprüfung) überprüft, ob die Konstrukte semantisch korrekt verwendet werden, und fügt semantische Informationen als Attribute in den Syntaxbaum ein.

generate : $AST' \rightarrow IntermediateCode$
Die Zwischenkode-Erzeugung (Generierung) erzeugt aus dem attributierten Syntaxbaum einen Zwischenkode, der vom Backend weiterverarbeitet wird.

Mit den beschriebenen Funktionen und einem Quelltext (*source*) als Eingabe läßt sich der (logische) Aufbau des Compilers funktional folgendermaßen beschreiben:

$$backend(frontend(source)) = backend(generate(check(abstract(parse(scan(source))))))$$

Diese Beschreibungsform betont, daß die einzelnen Phasen (Funktionen) nur vom Ergebnis der vorherigen Phase abhängen.

2 Beschreibungstechniken und Generatoren

Eine Programmiersprache kann durch passende Beschreibungstechniken formal spezifiziert werden, was sich als vorteilhaft für die Kommunikation zwischen Sprachdesignern und Compilerbauern sowie Anwendungsprogrammierern erweist. Generatoren können aus bestimmten Beschreibungen automatisch entsprechende Teile eines Compilers erzeugen. Wenn die Beschreibung eines Frontends vorliegt und entsprechende Generatoren zur Verfügung stehen, kann die Sprache geändert oder erweitert werden, indem die Beschreibung geändert und ein neues Frontend generiert wird, was weitaus einfacher und weniger fehlerträchtig als die Änderung eines handgeschriebenen Compilers ist.

2.1 Lexikalische Analyse

Die Abbildung von Zeichenfolgen auf Symbolfolgen wird durch reguläre Ausdrücke spezifiziert. Jedem Symbol der Sprache wird ein regulärer Ausdruck zugeordnet, der die Repräsentation des Symbols als Zeichenkette beschreibt. Umgekehrt muß jeder Zeichenkette, die in einem Quelltext auftauchen kann, ein Symbol der Sprache zugeordnet werden. Aus diesen Paaren (regulärer Ausdruck, Symbol) erzeugt ein Scanner-Generator einen endlichen Automaten, der die Abbildung realisiert.

Die folgende Tabelle zeigt einige Beispiele für reguläre Ausdrücke:

"begin"	Schlüsselwort: **begin**
":"	Schlüsselwort: Doppelpunkt
[0–9]+	ganze Zahl: mindestens eine Ziffer
[a–zA–Z][a–zA–Z0–9]*	Bezeichner: Buchstabe gefolgt von Buchstaben und Ziffern

Ein Scanner übernimmt weitere Aufgaben, die sich teilweise nicht durch reguläre Ausdrücke beschreiben lassen, beispielsweise das Entfernen von Leerzeichen, Zeilenumbrüchen und Zeilenvorschüben, Tabulatoren und (evtl. geschachtelten) Kommentaren, oder das Erkennen von speziellen Kommandos (*pragmas*), die den Compilerlauf beeinflußen, aber nicht zur Sprache gehören. Für diese Aufgaben müssen spezielle Beschreibungsformen eingeführt werden, worauf an dieser Stelle nicht näher eingegangen wird, da keine allgemein akzeptierte Beschreibungsform existiert.

Der bekannteste Scanner-Generator ist LEX, dessen Verwendbarkeit allerdings stark an das Betriebssystem UNIX und die Programmiersprache C gebunden ist.

2.2 Syntaktische Analyse

Die in einer Sprache erlaubten Symbolfolgen (die Sprachsyntax) werden durch kontextfreie Grammatiken beschrieben. Aus einer kontextfreien Grammatik erzeugt ein Parser-Generator einen Parser, der die eine Symbolfolge akzeptierenden Produktionen als Parsebaum darstellt.

Die grundlegende Form der kontextfreien Grammatiken beinhaltet folgende Elemente:

Terminal: Ein Symbol aus der Sprache, das vom Scanner erkannt wird, notiert als Zeichenkette oder Name eines im Scanner definierten regulären Ausdrucks.

Die folgende Tabelle zeigt einige Beispiele für Terminale:

"for"	das Schlüsselwort **for**
":"	ein Doppelpunkt, der auch als Schlüsselwort betrachtet wird
int	eine Zahl gemäß der Definition des Scanners
id	ein Bezeichner gemäß der Definition des Scanners

Nonterminal: Name einer Produktion. Beispiele: *ideG, valG.*

Produktion: Benannte Folgen von Terminalen und Nonterminalen, wobei auch leere Folgen möglich sind, die durch das leere Wort ϵ notiert werden. Jede Folge wird **Alternative** genannt.

Das folgende Beispiel zeigt eine Produktion mit vier Alternativen:

valG ::= "nil" | *int* | *id* | *valG* "+" *valG*

Der Name (Nonterminal) der Produktion lautet *valG*. Die akzeptierten Symbolfolgen (Alternativen) sind das Schlüsselwort nil, eine Zahl *int* oder ein Bezeichner *id* gemäß der Definition des Scanners sowie die Folge [*valG*, +, *valG*], wobei das Nonterminal *valG* rekursiv eine von der Produktion *valG* akzeptierte Symbolfolge beschreibt.

Kontextfreie Grammatiken können mehrdeutig sein, d. h. zu einer Symbolfolge lassen sich evtl. mehrere Parsebäume konstruieren, die diese Symbolfolge akzeptieren. Mit den verschiedenen Parsebäumen sind normalerweise auch verschiedene Semantiken verbunden, ein Programm sollte aber *eine* festgelegte Semantik besitzen. Deshalb schränken Parser-Generatoren die Klasse der erlaubten Grammatiken ein [Aho 88, Kap. 4] und/oder benötigen zusätzliche Informationen zur Auflösung von Mehrdeutigkeiten [Aho 88, Kap. 4.8]. Ein Generator kann also eine kontextfreie Grammatik als unzulässig zurückweisen. Verschiedene Generatoren können die gleiche Grammatik als zulässig oder unzulässig betrachten oder verschiedene Zusatzinformationen benötigen.

Zur Konstruktion von Parsebäumen sind zwei Verfahren möglich: Beim Top-Down-Verfahren wird der Parsebaum von der Wurzel (einer bestimmten Startproduktion) zu den Blättern (Symbolen) konstruiert. Bottom-Up-Parser konstruieren den Baum von den Blättern ausgehend zur Wurzel. Zu beiden Verfahren existieren Klassen von Grammatiken, die das Generieren eines deterministischen Parsers erlauben, der nach dem jeweiligen Verfahren den Parsebaum konstruiert. Bei der Entscheidung für eine Klasse von Grammatiken sind zwei sich widersprechende Aspekte zu bedenken: Die Menge der erlaubten Grammatiken ist bei Generatoren, die Bottom-Up-Parser z. B. aus LR-Grammatiken erzeugen, am größten. Nach dem Top-Down-Verfahren arbeitende Parser, die z. B. aus LL-Grammatiken erzeugt werden, sind (meistens) effizienter. Im Sinne der Allgemeinheit ist ein Generator für Bottom-Up-Parser zu empfehlen [Aho 88, S. 234].

Der bekannteste Parser-Generator ist YACC, dessen Verwendbarkeit allerdings stark an das Betriebssystem UNIX und die Programmiersprache C gebunden ist.

2.3 Abstraktion

Abstrakte Syntaxbäume werden durch Konstruktoren beschrieben, die als Parameter die Werte der Unterbäume bzw. Attribute des Knotens erhalten. Verschiedene Konstruktoren werden zu einer Sorte von Knoten zusammengefaßt.

Das folgende Beispiel zeigt Konstruktoren für abstrakte Syntaxbäume einer Sorte *val*:

```
val  ::=  nilNode()
        |  intNode(int)
        |  idNode(id)
        |  addNode(val val)
```


Ein Syntaxbaum der Sorte *val* ist entweder ein leerer Knoten *nilNode*, ein Knoten *intNode* mit einer ganzen Zahl *int* als Attribut, ein Knoten *idNode* mit einem Bezeichner *id* oder ein Knoten *addNode* mit zwei Unterbäumen der Sorte *val*.

Die Konstruktion eines abstrakten Syntaxbaums aus dem Parsebaum läßt sich durch semantische Aktionen in den Produktionen der kontextfreien Grammatik darstellen, wie das folgende Beispiel zeigt:

```
valG  ::=  "nil"              => nilNode()
        |  int                => intNode(int)
        |  id                 => idNode(id)
        |  valG1 "+" valG2    => addNode(valG1 valG2)
```

Durch das Schlüsselwort nil wird ein leerer Knoten *nilNode* erzeugt. Eine Integer-Zahl erzeugt einen Knoten *intNode*, der eine vom Scanner gelieferte Repräsentation *int* des Zahlenwertes als Attribut enthält. Ein Knoten *idNode* wird durch einen Bezeichner *id* erzeugt. Die Addition zweier Werte wird durch den Knoten *addNode* dargestellt, der als Unterbäume die von der Produktion *valG* erzeugten Syntaxbäume enthält.

2.4 Semantische Analyse

2.4.1 Attributierte Grammatiken

Die Analyse der statischen Semantik läßt sich mit attributierten Grammatiken beschreiben [Aho 88, Kap. 5]. Jedem Knoten des Syntaxbaums werden weitere Attribute zugeordnet und Vorschriften zur Berechnung dieser Attribute angegeben.

Das folgende Beispiel zeigt eine attributierte Grammatik mit Typinformationen:

```
val.env  ::=  nilNode()          =>  val.type := undefined
            |  intNode(int)      =>  val.type := integer
            |  idNode(id)        =>  val.type := lookup(val.env id)
            |  addNode(val1 val2) =>  val1.env := val.env;
                                      val2.env := val.env;
                                      if val1.type = integer
                                      andif val2.type = integer
                                      then val.type := integer
                                      else val.type := undefined
                                      endif
```

Jeder Knoten des Syntaxbaums der Sorte *val* erhält zwei Attribute: *env* ist eine Umgebung, in der jedem deklarierten Bezeichner ein Typ zugeordnet wird; *type* ist der Typ eines Knotens mit den möglichen Werten *undefined* und *integer*. Ein Knoten *nilNode* hat den Typ *undefined*, ein Knoten *intNode* den Typ *integer*. Der Typ eines Knotens *idNode* wird durch eine Funktion *lookup()* bestimmt, die den einem Bezeichner *id* in der Umgebung zugeordneten Typ zurückliefert. Der Typ des Knotens *addNode* ist *integer*, wenn beide Unterbäume den Typ *integer* haben, ansonsten

ist der Typ *undefined*. Außerdem könnten noch Fehlermeldungen erzeugt werden, was hier nicht gezeigt wird.

Die Attribute eines Knotens lassen sich in ererbte (*inherited*) und abgeleitete (*derived*) Attribute aufteilen. Ererbte Attribute eines Knotens erhalten ihren Wert durch einen Vorgängerknoten. Dieser Wert ist entweder der Wert eines ererbten Attributs des Vorgängerknotens oder der Wert eines abgeleiteten Attributs eines Knotens, der im Vorgängerknoten parallel zu diesem Knoten steht. Im Beispiel ist *val.env* ein ererbtes Attribut, dessen Wert im Knoten *addNode* an die Nachfolgerknoten weitergereicht wird. Die Werte der abgeleiteten Attribute werden durch die Nachfolgerknoten bestimmt. Im Beispiel ist *val.type* ein abgeleitetes Attribut.

Durch die Berechnungsvorschriften für die Attributwerte ergibt sich ein Abhängigkeitsgraph, der festlegt, wie oft bzw. in welcher Reihenfolge der Baum durchlaufen werden muß, um alle Attributwerte zu berechnen. Wünschenswert ist die Berechnung aller Attributwerte beim einmaligen Durchlauf durch den Baum, z. B. parallel zur syntaktischen Analyse. Dies wird durch durch links-attributierte Grammatiken sichergestellt [Aho 88, Kap. 5.4]. Bei links-attributierten Grammatiken hängt der Wert eines ererbten Attributs nur von den ererbten Attributen der Vorgängerknoten und den abgeleiteten Attributen der im Vorgängerknoten links von diesem Knoten stehenden Knoten ab. Im obigen Beispiel dürfte im Knoten *addNode* das ererbte Attribut $val_2.env$ von dem abgeleiteten Attribut $val_1.type$ abhängig sein, aber nicht umgekehrt $val_1.env$ von $val_2.type$.

Links-attributierte Grammatiken korrespondieren sowohl mit LL-Grammatiken und Generatoren für Top-Down-Parser als auch mit LR-Grammatiken und Generatoren für Bottom-Up-Parser.

2.4.2 Prädikate

Zur Analyse der statischen Semantik können Regeln formuliert werden, die festlegen, welche abstrakten Syntaxbäume wohlgeformt sind. Ein automatischer Beweiser (oder ein Prolog-Interpreter) kann mit Hilfe der Regeln die Wohlgeformtheit eines Syntaxbaums überprüfen.

Das folgende Beispiel zeigt Regeln für wohlgeformte Syntaxbäume:

[IntNode]

$$\frac{}{Env \vdash intNode(int) : Int}$$

[IdNode]

$$\frac{id \notin Env'}{(Env, id : Int, Env') \vdash idNode(id) : Int}$$

[AddNode]

$$\frac{Env \vdash val_1 : Int \quad Env \vdash val_2 : Int}{Env \vdash addNode(val_1\ val_2) : Int}$$

Für Syntaxbäume *v* der Sorte *val* wird ein Prädikat $v : Int$ definiert. Das Prädikat wird von jedem Knoten *intNode* erfüllt, wie die Regel [IntNode] angibt. Ein Knoten *idNode* erfüllt das Prädikat, wenn in der Umgebung dem Bezeichner *id* der Typ *Int* zugeordnet ist und der Bezeichner im Rest der Umgebung *Env'* nicht redefiniert wird. Gemäß der Regel [AddNode] erfüllt ein Knoten *addNode* das Prädikat, wenn beide Unterbäume in der gleichen Umgebung *Env* das Prädikat erfüllen.

2.5 Zwischenkode-Erzeugung

Die Erzeugung des Zwischenkodes aus dem Syntaxbaum läßt sich wie die Typprüfung durch attributierte Grammatiken beschreiben [Aho 88, Kap. 8], wie das folgende Beispiel zeigt:

val.env	::=	*nilNode()*	=> *val.code := nop*
	|	*intNode(int)*	=> *val.code := push(int)*
	|	*idNode(id)*	=> *val.code := load(val.env id)*
	|	*addNode(val_1 val_2)*	=> *val.code := seq(val_1.code val_2.code add)*

Jeder Knoten des Syntaxbaums erhält ein weiteres Attribut *code*, das die Werte *nop* (keine Operation), *push(int)* (lege die Zahl *int* auf dem Keller ab), *load(env id)* (lege den durch den Bezeichner *id* in der Umgebung *env* benannten Wert auf dem Keller ab), *add* (addiere die beiden obersten Elemente des Kellers und lege das Ergebnis auf dem Keller ab) und *seq(x_i)* (führe die Operationen x_i nacheinander aus) annehmen kann. Der Knoten *nilNode* erzeugt den Befehl *nop* und der Knoten *intNode* den Befehl *push* mit dem Zahlenwert-Attribut als Argument. Der Knoten *idNode* führt dazu, daß ein durch den Bezeichner *id* benannter Wert auf dem Keller abgelegt wird. Der Knoten *addNode* erzeugt eine Befehlssequenz aus den Befehlen der beiden Unterbäume und dem Befehl *add*.

3 Schluß

Ich habe in diesem Artikel einige (formale) Beschreibungen für Teile eines Compilers vorgestellt. Auch ohne die entsprechenden Werkzeuge sind diese Beschreibungen sinnvoll, um einen Compiler methodisch und gut dokumentiert zu entwickeln. Mit den entsprechenden Werkzeugen wird die Entwicklung eines Compilers einfacher und sicherer. Der resultierende Compiler ist änderungsfreundlicher und wartbarer als sein handgeschriebenes Gegenstück.

Wenn Ihr Fragen, Anregungen, Wünsche für Folge-Artikel o.ä. habt, sagt mir Bescheid!

Literatur

[Aho 88] A. V. Aho, R. Sethi und J. D. Ullman: *Compilerbau*. Addison-Wesley 1988.

[Waite 85] W. M. Waite und G. Goos: *Compiler Construction*. Springer 1985.

[Wirth 86] N. Wirth: *Compilerbau*. Teubner 1986.

ZCPR - eine Allzweckwaffe?

Zwischen Benutzern des Z-Systems und Leuten, die damit noch keine nähere Bekanntschaft gemacht haben, gibt es immer wieder Diskussionen über seine Vor- und Nachteile. So auch in unserer Club-80 Ecke in der Discovery-Mailbox.

Fritz Chwolka stellte dort beispielsweise die provozierende Frage: "Welchen Sinn hat denn ein Z80-Rechner ohne ZCPR ?!" Und Alexander Schmid schrieb: "Wie mir Helmut Jungkunz vor kurzem gesagt hat, hat er die Preise für die selbstinstallierenden ZCPR-Pakete ("NZCOM" für CP/M 2.2 und "Z3PLUS" für CP/M 3) auf jeweils 70 Mark gesenkt. Für diesen Preis dürfte es aber wohl einmalig sein, denn im Preis inbegriffen ist eine kostenlose Versorgung mit Public Domain Software zum Z-System, die momentan über 50 MB umfasst. Nein, ich bekomme keine Prozente, aber ich arbeite selber mit dem Z3PLUS und ich würde mich mit Händen und Füssen wehren, wenn ich es wieder hergeben sollte."

Als langjähriger Besitzer eines Commodore 128 D, der - ausser im Urlaub - keinen Tag ungenutzt herumsteht, schaue ich mich natürlich ständig nach neuen Programmen um. Warum sollte neben Verbesserungen bei Anwender-Programmen nicht auch eine Betriebssystem-Verbesserung möglich sein? ZCPR steht ja für "Z Console Processor Replacement" und soll den standardmässig zum CP/M-System gehörenden "Console Command Processor" (CCP) ersetzen. Ursprünglich lag wohl der Gedanke zugrunde, dass das für den Intel 8080 Prozessor-Chip entwickelte CP/M-Betriebssystem eigentlich mehr könnte, wenn man es an die Zilog Z-80, -180 und -280 Prozessoren der moderneren CP/M-Maschinen anpassen würde, die seit etwa 1980 auf dem Markt erschienen.

Meine genaueren Nachforschungen ergaben dann auch, dass Leute, die von ZCPR so begeistert sind, hauptsächlich mit "aufgebohrten" modernen CP/M-Maschinen arbeiten. Fritz beispielsweise ist stolzer Besitzer eines kleinen Museums mit an die 30 verschiedenen CP/M-Computern, darunter auch dem legendären Commodore PET 2001 mit Diskdrive, sowie einem C-128. Sein Arbeitsgerät ist jedoch ein Z-280 Rehdesign mit 60 MB Harddrive, IDE-Interface und Terminalkarte von Dipl.-Ing. Tilmann Reh, Siegen. Ebenso lüftete Alexander - einer der wenigen Profis, die privat noch unter CP/M programmieren - das Geheimnis seiner Anlage etwas, als er mir mitteilte: "Ich habe hier einen Z-280-Rechner mit 12,5 MHz und einer 80 MB Festplatte, und da kann ich fast beliebig viele Utilities zur ständigen Verfügung halten. Und wenn Programme wie der WordStar Overlays nachladen, zuckt bei mir auch beim schnellen Tippen nur kurz der Cursor. Auf einem 2 MHz Z-80 mit 140 KB pro Laufwerk kann man das wohl vergessen." - Wer jetzt übrigens noch rätselt, was Helmut Jungkunz wohl für einen Rechner einsetzt, dem kann schnell geholfen werden: ein CPU 280!

Die Leistungsmerkmale meines C-128 - Z-80 Prozessor mit 4 MHz Taktgeschwindigkeit - sind dagegen also bescheiden, wobei von den 128 KB Hauptspeicher nach Laden des Standard-CP/M-Plus nur noch 58 KB Arbeitsspeicher (TPA) zur Verfügung stehen. Und wo ausser dem eingebauten Laufwerk für zweiseitige 5 1/4"-Disketten (mit 300 - 400 KB) allenfalls weitere 5 1/4"- oder 3 1/2"-Laufwerke extern angeschlossen werden können, oder eine RAM-Disk mit 508 KB.

Derartige Unterschiede in der Hardware bedingen notwenigerweise aber auch Unterschiede in der Software. So benötigt CP/M 3 - obwohl es theoretisch auch für den alten 8080 Prozessor geeignet ist - in der Praxis mehr Speicherplatz und damit eben auch einen schnelleren Prozessor, als das ältere CP/M 2. Und ebenso ist auf den verschiedenen PC-DOSen ja auch nicht in allen Fällen das gleiche Betriebssystem im Einsatz.

Um nicht erst nach erfolgtem Kauf und Hereinfall zu wissen, dass eine eventuelle Anschaffung des ZCPR für meinen 128-er vielleicht nur hinausgeschmissenes Geld ist, nahm ich also die Herausforderung an und teilte Alexander meine Meinung mit. Nachstehend einige Auszüge aus meinem Mailbox-Dialog mit ihm:

GB> Es ist schon interessant, dass die ZCPR-Autoinstaller
GB> inzwischen von ehemals DM 150,-- auf DM 70,-- gefallen
GB> sind. Als gelernter Kaufmann wuerde ich das allerdings
GB> mangelnder Nachfrage zuschreiben. Und da die dazu passende
GB> P-D-Software keineswegs kostenlos ist - das Einschicken
GB> formatierter Disketten mit Rueckumschlag und Rueckporto
GB> finde ich sogar ziemlich laestig -, wuerde ich "Z3PLUS" z.
GB> Z. nicht einmal einfuehren, wenn das Install-Paket
GB> kostenlos waere: Meine bisherige Marken-Software (ausser
GB> "BDS-C" und "Wordstar 4.0") wird davon ja nicht besonders
GB> unterstuetzt. Und bei ZCPR-Programmen haette ich
GB> Kompatibilitaetsprobleme beim Dikettentausch mit Kollegen,
GB> die kein ZCPR haben! Aber vielleicht veroeffentlichst Du
GB> hier gelegentlich mal eine Liste der Software, mit der Du
GB> unter "Z3PLUS" arbeitest? Eventuell koennte man darunter
GB> etwas Interessantes entdecken?

Darauf reagierte Alexander zunächst zwar etwas sauer. Er (wie übrigens auch Fritz) ist immerhin Software-Verteiler der amerikanischen ZSIG-Organisation unter Helmut Jungkunz, der wiederum seit Frühjahr 1990 den Alleinvertrieb für Deutschland übernommen hat.

Wenn Diskussionspartner aber ehrlich auf der Suche nach dem richtigen Standpunkt und der Wahrheit sind, anstatt mit Gewalt rechthaben zu wollen, dann lässt sich auch ein Kompromiss finden.

GB> Wie mir scheint, ist ZCPR softwareseitig der Bereich,
GB> der mithilft, Deine leistungsfähige Hardware zu einer
GB> überdurchschnittlichen Z280-Anlage zu machen. Bei langsamen
GB> Rechnern mit geringer Speicherkapazität lohnt sich der
GB> Aufwand (der ZCPR-Installation) dagegen möglicherweise
GB> tatsächlich nicht; also beispielsweise kein CPC 664 ohne
GB> Speichererweiterung und zusätzlichem 5 1/4"-Laufwerk.

(Anmerkung: Der Schneider CPC 664 hatte einen Hauptspeicher von nur 64 KB für Betriebssystem und Anwenderprogramm zusammen, sowie ein Laufwerk für einseitige 3"-Disketten mit etwa 180 KB.)

AS: Dem kann ich nichts mehr hinzufuegen. Windows auf einem 8086er wuerde ja sicher auch keinen Spass machen ...

GB> Natürlich möchte ich Dich keineswegs drängeln, über Deine
GB> rund 150 ZCPR-Utilities jeweils einen Aufsatz zu schreiben,
GB> aber die wichtigsten dieser Programme wie Dein ZFILER
GB> würden mich schon näher interessieren.

AS: Der ZFILER selber bietet eigentlich recht wenige Funktionen, er ist mehr der Verwalter einer maechtigen Macrobibliothek. Das Hilfsmenue des ZF sieht so aus:

```
                    -- Filer Commands --
    A - Alpha Sort  C - Copy    D - Delete  J - Jump     T - Tag
    F - File Size   M - Move    P - Print   R - Rename   U - Untag
    W - Wild Tag    V - View    Y - Retag                 View
G - Group: Archive Copy Delete Fsize Move Print Reverse Tag Untag

    -- Misc --                      -- Cursor --
                                ^R  - Top of Screen       WS diamond
X - eXit ZFILER                 ^C  - End of Screen
L - Login DU:/DIR:              ^T  - First File              ^E
S - disk Status                 ^B  - Last File               ^
Z - ZCPR3 command              +/^F - Next Screen         ^S <-+-> ^D
H - Help ZFILER                -/^A - Prev Screen             v
E - refresh scrEen           SP/CR  - Next File               ^X
O - Option toggles             BS   - Prev File
```

Die eigentliche Funktion besteht nun darin, dass man auf dem Monitor ein sortiertes Directory hat und darin mit dem Cursor beliebig rumfahren kann. Hat man das File ausgewaehlt, kann man es mit den Grundfunktionen kopieren, verschieben, ausdrucken oder umbenennen. Wenn das nicht reicht, kann man den Filenamen mit einem entsprechenden Macro mit einem einzigen Tastendruck an externe Programme uebergeben, die dann denken, man haette das in der Kommandozeile eingetippt. Wenn man einen Befehl auf mehrere Files anwenden will, kann man diese markieren und dann werden die der Reihe nach bearbeitet. Das ist z.B. beim Crunchen von mehreren Files sehr praktisch.

Andere Annehmlichkeiten des ZCPR werden zwar auch durch externe Programme realisiert, aber die haengen so eng mit dem System zusammen, dass sie ohne ZCPR einfach keinen Sinn ergeben. Wenn man im ZCPR z.B. einen Filenamen eingibt, den es auf dem eingeloggten Laufwerk nicht gibt, wird erstmal der interne Suchpfad abgeklappert und wenn da auch nichts gefunden wird, wird ein Programm namens CMDRUN gestartet, das dann die weitere Fehlerbehandlung uebernehmen kann. Im Normalfall durchsucht dieses Programm ein Textfile namens ALIAS.CMD, das folgendermassen aufgebaut ist:

```
brief      cd wordstar;ws $1
```

Wenn man also BRIEF GUENTHER eintippt und es kein BRIEF.COM gibt, findet CMDRUN im ALIAS.CMD obige Zeile, loggt im Directory WORDSTAR ein und ruft den Wordstar mit WS GUENTHER auf. Das File kann mehr oder weniger beliebig lang werden und wenn man will, kann man sich fuer fast alles eine Abkuerzung schreiben.

Oder man macht das System auch fuer DOSler benutzbar, indem man

z. B. schreibt:

```
era=erase=del=delete      era $1
```

Egal, was man dann von den Befehlen links eintippt, es wird das Gewuenschte getan, naemlich das File geloescht.

Wenn man da nun einen langsamen Rechner mit langsamen Laufwerken hat, wartet man sich tot, bis man wieder beim CP/M-Prompt landet und wird das Z-System sicher umgehend wieder rauswerfen.

Das war also die instruktive Stellungnahme Alexander Schmids: Für Leute mit entsprechenden Rechnern sicher ein Anlass zum Nachdenken, ob sie jetzt in diesem Winter nicht die Einführung des ZCPR in Angriff nehmen sollten.

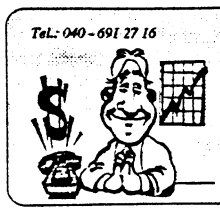
Für Commodore 128 User ergibt sich jedoch noch kein Handlungsbedarf. Dass andere C-128 Besitzer ebenso denken, beweist auch die Adressenliste von Helmut Jungkunz. Anfang 1993 hatten dort von knapp 100 eingetragenen ZCPR-Usern nur etwa 10 einen C-128. Für diesen Rechner wären einige andere Projekte von viel grösserem Interesse, beispielsweise:

8 MHz Karte: Die Firma Rossmöller Handshake GmbH in Meckenheim brachte bis 1990 eine "Turbo CP/M 128" Karte für DM 99,-- auf den Markt, womit der von Commodore eingebaute Prozessor Z-80 A durch einen Z-80 H ersetzt und die Rechengeschwindigkeit damit verdoppelt werden konnte. Wer Glück hat, kann sich vielleicht noch so ein Teil gebraucht beschaffen.

DCF-77 Zeit-Empfänger: Die Firma Conrad-Electronic in Hirschau brachte bis 1991 einen zündholzschachtel-grossen Atomuhr-Empfänger zum Preis von etwa DM 60,-- für den Userport des C-64 heraus (heute nur noch für Amiga?). Wenn jemand die dazugehörende Assembler-Software an das CP/M anpassen würde, wären auf dem C-128 automatische Zeiteinträge wie auf PCs möglich.

Festplatte: Verschiedene Firmen (wie die Scanntronik GmbH in Zorneding) liefern die amerikanische CMD-Harddisc für C-64 und C-128 mit 20 MB aufwärts und ab etwa 1100 Mark. Das Gerät wird von CP/M leider nicht optimal unterstützt und lässt sich bisher nur in Diskettengrössen partitionieren. Zum Einsatz dieser Festplatte wäre also eine Betriebssystem-Änderung durchzuführen.

Wie aus der Mitglieder-Umfrage im vergangenen Sommer ersichtlich wurde, besitzt über die Hälfte der Club-80 Mitglieder inzwischen (auch) einen PC und arbeitet mehr oder weniger häufig (nur) damit. Die Anwender von CP/M-Computern werden also immer weniger. Berücksichtigt man daneben aber auch die zusätzlichen Leser der Club-Informationen aus der Anwendergruppe "CP/M aktuell", so ist der Commodore 128 derzeit der noch meistbenutzte CP/M-Computer überhaupt. Und da anzunehmen ist, dass von diesen "Probe-Lesern" ab Januar 1994 doch einige ihren Beitrag auf das Konto des Club-80 überweisen, ist die Schlussfolgerung wohl erlaubt, dass wir hier eine Gruppe von C-128 Anwendern zusammenbekommen, unter denen vielleicht sogar ein ambitionierter System-Programmierer ist. Man darf gespannt sein! (GWB)



Tel.: 040 - 691 27 16

Jörg Brans - Tieloh 55, 22307 Hamburg

Club 80 Börse

Liebe Clubfreunde !

Da ich mich aus Zeitmangel nur noch mit MS-DOS beschäftigen kann, will ich mich daher von meinen Tandy Computern, Software usw. trennen. Vielleicht ist jemand im Club, der an dem folgendem Angebot Interesse hat:

Computer

Zwei Model 4P mit folgender Ausstattung:

1. Gerät 8 MHz (auf 2-4-6-8 per Software), 512K RAM, 2 x 40/80 Floppy - umschaltbar, Schnittstellen: P/S, Floppyanschluß extern

2. Gerät 6,3 MHz, 512K RAM, 2 x 40/80 Floppy - umschaltbar, Zeichensatz Tandy u. IBM (umschaltbar), Schnittstellen: P/S, Floppyanschluß extern

Peripherie

Gehäuse mit 2 Netzt. für 2 externe Laufwerke (eingebaut 1 x 3.5 Zoll - 720K Laufwerk)

Software (mit Lizenz)

Betriebssysteme : LSDOS 6.2 (deutsch) - LSDOS 6.3 (englisch)

Textverarbeitung: 1.) LeScript Vers. 2.0 (Spezialv. für deutsche Tastatur) mit zus. Druckertreiber für HP-Deskjet von Anitek
2.) Allwrite

Sprachen : 1.) Multi-Basic Vers. 1.00.01
2.) Bascom - Basic Compiler
3.) Pro-Create Editor/Assembler
4.) Assembly Language Development System (ALDS)

Integrierte Software : T/Maker mit folgenden 9 Funktionen:
* File Management
* Word Processing
* Spell Checking
* Speadsheet
* Database Management
* List Processing
* Data Transfer
* Graphics (Bar Charts)
* Programming

Utilities : Model 4 ToolBelt, Packer, DoubleDuty, Hyperdrive, Superdrive

Literatur : Mod 4 by Chris, Using Super Utility+ 3.x - Super Utility 4/4P
Job Control Language f. LDOS 5.3, TRSDOS 6.1 6.2 6.3
Programmierung des Z 80 v. Rodnay Zaks, ROM Listing Mod III
Vom Umgang mit CP/M, Multiplan deutsch, Schaltpläne Mod. 4/4P
Technical Reference Manual Mod. 4P, Micro 80 (letzte Jahrgänge)
Das DOS Buch (NEWDOS,GDOS,Color-DOS) f. TRS -80 I+III,
Genie I,II,III,IIs + IIIs, Color Genie
Basic Handbuch v. ITT Microcomputer Software

Ich bin mir nicht sicher, ob ich an alles gedacht habe. Aber jetzt will ich zum Preis kommen.

Meine Preisvorstellung ist 700,- DM zzgl. Versandkosten

Selbstabholer wäre natürlich ideal. Wenn Interesse besteht, erhält der Abnehmer noch zusätzlich kostenlos einen Drucker DMP 430/DIN A3 (Druckkopf defekt). Außerdem gehört noch jede Menge PD-Software zum Angebot.

Wer sich dieses einmalige Angebot entgehen läßt hat selbst schuld. Also greift zum Telefon und wählt 040 - 691 27 16 !!

Gruß an alle die ich kenne

Jörg

Deskjet News

Der HP Deskjet duerfte inzwischen ja recht weit verbreitet sein und neben den diversen Nachfuellrezepten fuer die Tintenpatronen habe ich vor kurzem einen anderen sehr interessanten Tip im FIDO gelesen.

Wenn man im Ausdruck immer wieder weisse Streifen hat, kann das entweder an einer eingetrockneten Duese, an Luftblasen, die beim Nachfuellen entstanden sind, oder an einer Unterbrechung der Stromzufuehrung einer Duese bzw. einem durchgebrannten Heizelement in einer Duese liegen. Wenn Die Streifen nur von Zeit zu Zeit auftreten, liegt es wahrscheinlich an Luftblasen und es hilft oft, wenn man die Patrone mit der Seite flach auf einen Tisch klopft. Um aber festzustellen, ob eine Patrone wirklich defekt, oder vielleicht auch nur nicht richtig eingerastet ist, braucht man keine teuren Messgeraete, das sagt einem freundlicherweise der Selbsttest, auch wenn ich dazu im Handbuch nichts gefunden habe. Man drueckt dazu waehrend des Einschaltens die FONT-Taste und beobachtet die Dinge, die sich entwickeln. Ganz am oberen Rand, noch vor dem Treppchen, steht normalerweise ein "ID H" o.ae., wenn dort aber Zahlen vor dem ID stehen, z.B. "11 24 ID E", dann bekommen in diesem Beispiel die Duesen Nr. 11 und 24 keinen Saft. Wenn man Glueck hat, hat man nur mit seinen Marmeladenfingern auf die Kontakte gefasst und es reicht, wenn man sie mit einem spiritusgetraenkten Lappen reinigt. Wenn dann auch ein vorsichtiges Festdruecken nicht hilft, ist wohl ein Heizelement durchgebrannt und man kann die Patrone entsorgen.

Wenn das dann mal der Fall sein sollte, steht man allerdings vor dem Problem, einen neuen Druckkopf zu bekommen. Immer mehr Haendler scheinen naemlich der Meinung zu sein, dass es fuer sie doch wesentlich guenstiger ist, wenn sie die Nachfuellsets gleich selber zu einem Wucherpreis verkaufen. In den meisten Geschaeften bekommt man dann nur noch diese Sets oder die Patronen mit der doppelten Kapazitaet, die man aber praktisch nicht nachfuellen kann. Bei den "normalen" Patronen heisst es dann immer, dass es die nicht mehr gibt. Irrtum, nur steht vorne auf der Schachtel DeskJet 300J und DeskJet Portable. Ist aber die alte Patrone und hinten drauf steht auch die ganze Litanei an Druckern, wo die Patrone sonst noch reinpasst. Also, nicht verar***** lassen, die Patronen fuer den tragbaren DJ kaufen und munter weiter selber nachfuellen.

Alexander Schmid

— FMail 0.94+
* Origin: * Discovery Club 80 - Ecke * LifeNet (77:7700/740)

MODEM umsonst!!!???

Schon im letzten Info habe ich kurz die wohl preiswerteste Möglichkeit angesprochen, zu einem postzugelassenen Modem zu kommen. Sicher habt ihr aber auch schon selbst die Werbung der 1&1 Telekommunikation GmbH gesehen, die in letzter Zeit des öfteren Computerzeitschriften beigelegt war.

Ein Modem zum Nulltarif, nämlich auf unbegrenzte Zeit kostenfrei ausgeliehen, wird da angeboten. Verbunden mit dem Leihmodem ist zudem eine kostenlose Eintragung als BTX-Teilnehmer, die normalerweise schon 65,– DM kostet. Und obendrein gibt es ein, ebenfalls kostenloses, BTX-Decoder-Programm für PC's (DOS oder WINDOWS), Atari oder Amiga.

Aber was bringt das Angebot dem Besitzer eines CP/M-Rechners, der nur auf der Suche nach einem billigen Modem für den Einstieg in die DFÜ ist? Leider nichts!

Das von 1&1 kostenfrei abgegebene Modem ist ein speziell für den BTX-Anschluß nach der "alten" V.23-Norm konzipiertes Gerät. Es arbeitet deshalb auch nur mit den, im restlichen DFÜ-Bereich (zum Glück) unüblichen, Übertragungsraten von 75 bps für den Sende- und 1200 bps für den Empfangsbetrieb. Diese Übertragungsgeschwindigkeit ist nicht, wie z.B. bei hayeskompatiblen Modems üblich, vom Rechner aus änderbar. Damit ist das Modem für DFÜ-Einsteiger aus dem CP/M-Lager praktisch wertlos.

Im Gegensatz dazu sind die ebenfalls zu einem recht günstigen, allerdings inzwischen fast marktüblichen Kaufpreis angebotenen 2400'er (129,–) und 14400'er (349.–) Modems der gleichen Firma durchaus auch für andere Zwecke als den BTX-Anschluß verwendbar. Auch in diesem Preis ist die BTX-Anschlußgebühr von 65,– DM enthalten. Stellt sich nur die Frage, welchen Wert der BTX-Anschluß für einen CP/M-User hat, für dessen Rechner es keinen Software-Decoder gibt (oder sollte da etwas an mir vorbeigelaufen sein)?

Und so interessant ist BTX nun auch wieder nicht, daß man sich dafür extra einen PC zulegen müßte ;-)

Allways good Hacking, Mt+

Impressum

1. Vorsitzender: **Hartmut Obermann** Tel.: 0 82 21/ 3 02 48
Mozartring 23 BTX: 0 82 21/ 3 02 48
Postfach 14 30 FAX: 0 82 21/ 3 35 75
89304 Günzburg

2. Vorsitzender: **Gerald Schröder** Tel.: 0 41 05/ 26 02
Am Schützenplatz 14
21218 Seevetal

Hardwarekoordinator: **Andreas Magnus** Tel.: 02 09/ 87 02 30
Bismarckstraße 29
45879 Gelsenkirchen

NewDOS-Diskothekar: **Oliver Volz** Tel.: 07 11/ 74 40 51
Am Ochsenwald 37A
70565 Stuttgart (Rohrerhöhe)

CP/M-Diskothekar: **Fritz Chwolka** Tel.: 0 24 64/ 89 20
Saarstraße 34
52457 Aldenhoven

C-128-Diskothekar: **Günther W. Braun**
Postfach 80 02 26
81602 München

Clubbücherei: **Kurt Müller** Tel.: 0 41 52/ 7 06 43
Sophie-Scholl-Ring 3b
21502 Geesthacht

Redaktion: **Jens Neueder** Tel.: 07 91/ 4 28 77
Gschlachtenbretzingen BTX: 07 91/ 44 47 22
Rudolf-Then-Straße 32 FAX: 0 79 71/ 2 50 55
74544 Michelbach / Bilz

Bankverbindung: **Club 80** **Postgiroamt Frankfurt**
Postgiro Sonderkonto CLUB 80 **BLZ: 500 100 60**
Obermann H., 8870 Günzburg **Kto.Nr.: 496 071 - 605**

Autoren: Die Redaktion bedankt sich bei den im **Inhaltsverzeichnis** genannten Autoren für die Mitarbeit an der Club-INFO. Eine Zensur oder Kontrolle der INFO-Beiträge erfolgt nicht.

Schluß

Hallo Club 80'er,

zum Jahreswechsel haltet Ihr das 42. Club-Info in den Händen. Ich möchte mich an dieser Stelle für Eure Mitarbeit an unserer Club-Info bedanken und für die Feiertage die besten Wünsche übermitteln.

Im Anhang des Info's ist diesmal dabei:

InfoForm - Info & Tips für Artikelschreiber
Messeliste - Übersicht diverser Computermessen
Geburtstagsliste - Aufstellung der Geburtstage der Clubmitglieder

Zu der Geburtstagsliste möchte ich noch anmerken, daß sie nicht vollständig ist, da nicht alle Clubmitglieder Ihr Geburtsdatum angegeben haben. Hervorzuheben sind vielleicht die Häufung der Geburtstage im April sowie die Tage **19.04.** und **31.05.** Obwohl wir eine kleine Gruppe sind, und es doch über 360 Möglichkeiten gibt einen Geburtstag zu feiern, haben sich zu beiden Tagen jeweils drei von uns entschieden gemeinsam Geburtstag zu feiern. Das dann noch ein genau 10-jähriger Altersunterschied hinzukommt ist sicher nur Zufall. Sicher sind diese zwei Termine besonders geeignet für Clubtreffen, da die "**Rundenzahler**" schon feststehen. ...soweit zum Mißbrauch von Statistiken.

Ich hoffe die Winter-/Schlechtwetterzeit ermöglicht Euch die Ausarbeitung neuer Artikel für unsere Club-Info. Viel Spaß beim Computern. Ich wünsch Euch nochmals ein Gutes Neues.
Bis zum nächsten mal Euer

Jens

lfd. Nr.	Nachname	Vorname	Straße		PLZ	Ort	Telefon privat Tel. geschäftl.	Telefax privat FAX geschäftl.	BTX FIDO-Node	Mailboxname Mailboxnummer
26	Mössel	Franz	Schafferstraße 12	I	39012	Meran	0039-473/34178 0039-471/980496	- -	- 2:333/400	- -
	IBM, Joyce, Workemate-Bullet▪ Modem									
27	Neueder	Jens	Rudolf-Then-Straße 32	GER	74544	Michelbach / Bilz	0791/ 42877 07971/ 250-50	- 07971/ 250-55	791444722 1001667	CCWN 0715168434
	IBM 286/386/486, Atari ST 1040, TRS80 MI▪ Modem, Sound Galaxi NX, Scanner, Tape, Borsu									
28	Neumann	Christof	Zeitblomstraße 22/2	GER	89077	Ulm /Donau	0731/ 6022568 0731/ 9749720	- -	- -	- -
	IBM, Tandy MII, Tandy M4p▪ Novell									
29	Obermann	Hartmut	Mozartring 23	GER	89312	Günzburg	08221/ 30248 -	08221/ 33575 -	0822130248 2:241/7922.10	SYNREL 08282/ 4311
	IBM486, Tandy M4p, Epson PX-8 ▪ Modem, Scanner									
30	Retzlaff	Bernd	Kleiner Sand 98	GER	25436	Uetersen	04122/ 43551 04103/ 605310	- -	- -	- -
	IBM 386, C64, GENIE I▪									
31	Rinio	Gerd	Rennbahnstraße 9	GER	22111	Hamburg	040/6552630 -	- -	- -	- -
	IBM 486DX66/2, RTS 80, TRS 80 RS ▪ Modem									
32	Ruschinski	Claus	Pommernstraße 21	GER	45770	Marl	02365/ 34646 -	- -	- -	- -
	IBM 386, TRS80 M I▪ Highscreen-Scanner, CoProz IIT387									
33	Schimmer	Jörg	Stettinerstraße 28	GER	60388	Frankfurt	06109/ 35336 069/ 3800-2385	- -	- 2:249/70.9	- -
	IBM 486, Schneider CPC▪ Modem CSR2400									
34	Schmid	Alexander	Entmannsdorf 5	GER	96317	Kronach /Gehülz	09261/ 53496 -	- -	- 2:2400/830	- -
	GENIE IIs, GENIE IIIs, CPU 280▪ Modem 2400,Prommer80,Ramdisk,Club80Terminal									
35	Schmitz	Rainer	Küferweg 12/1	GER	73099	Adelberg	07166/ 1397 07161/ 608-475	- -	- -	- -
	IBM,Portfolio,NCR Decision Mate V,Joyce,... ▪ Modem 1200, Märklin Digital Interface 6050									
36	Schoberth	Uwe	Petrus-Waldus-Straße 14	GER	75443	Oetisheim	07041/ 7254 0711/ 89394500	- 0711/ 89394513	- -	- -
	Alphatronic P3▪									
37	Scholz	Hans-Werner	Spitalstraße 54	GER	41334	Nettetal	02157/ 3613 -	- -	- -	- -
	IBM 386, Prof 80, ITT 3030▪ Prommer80									
38	Schroers	Horst-Dieter	Breslauer Straße 9	GER	85622	Feldkirchen	089/ 9032615 -	089/ 9043413 -	- -	- -
	IBM▪ Modem, Scanner									
39	Schröder	Gerald	Arminiusstraße 2	GER	22525	Hamburg	040/ 8507131 040/ 54715334	- -	- -	- -
	IBM 386SX, Atari 1040STF, Z280▪									
40	Schröer	Egbert	Joachimstraße 18	GER	46284	Dorsten	02362/ 75311 02362/ 49-9649	- -	- -	- -
	Portfolio, TRS80 MI, GENIE I, GENIE IIIs▪									
41	Schulte	Hartmut	Entenschnabel 8	GER	31311	Uetze	05173/ 1248 -	05173/ 24631 -	- -	- -
	IBM, Z280 u.a.▪ Scanner									
42	Sonnemann	Harald	In den Eckwiesen 9	GER	64405	Fischbachtal	06166/ 8512 06151/ 92-1265	- -	- -	- -
	NDR-Klein ▪ parallel/seriell, EPROMer									
43	Stumpferl	Stefan	Hasenberglstraße 57	GER	80933	München	089/ 3138193 -	//0893144001// -	- -	- -
	Amstrad CPC 6128+, GENIE IIIs ▪ Modem, 20MB-Wechselplatte, PSG&PIO,SIO, ...									
44	Sörensen	Rüdiger	Wiesbadener Str. 288	GER	55252	Mainz-Kastel	06134/65342 -	- -	- -	- -
45	Tornow	Wilhelm	Elbblick 46	GER	21629	Neu Wulmstorf	040/ 7007280 -	040/ 7003854 -	- -	- -
	IBM 386DX, Atari Mega ST4, Tandy M4p▪ Streamer,Soundblaster,Modem 2400,F.A.K.S. 910									
46	Vogl	Michael	Weidenweg 15	GER	41515	Grefenbroich /Laach	02181/45112 -	- -	0218145112-0001 -	- -
	IBM, Amstarad CPC 464/6128▪ Modem Gigitek 2400, Scanner Dat F.CPC									
47	Volkmer	Richart	Am Spörkel 69	GER	44227	Dortmund	0231/ 752574 -	- -	- -	- -
	IBM XT, Apple LC, CPC 6128, Osborne, TandyMII▪ Modem 2400, ScanMan-Scanner									
48	Volz	Oliver	Am Ochsenwald 37A	GER	70565	Stuttgart (Rohrerhöhe)	0711/ 744051 0711/ 685-3013	- -	- -	- -
	IBM, GENIE IIs▪ Modem Avantec									
49	Werner	Heiko	Reichenberger Straße 5	GER	01129	Dresden	0351/ 4608612 -	- -	- -	- -
	IBM 286/486▪ Modem 2400, Scanner									

Adressen-Liste CLUB80

Stand vom 29-12-1993
Seite 1

lfd. Nr.	Nachname	Vorname	Straße	PLZ	Ort	Telefon privat Tel. geschäftl.	Telefax privat FAX geschäftl.	BTX FIDO-Node	Mailboxname Mailboxnummer
1	Barendt	Harry	Hermann-Löns-Straße 7	GER 50181	Breedburg (Erft)	02272/7168 -	- -	- -	- -
2	Berndt-Jochum	Ilse	Stachelsgut 24	GER 51427	Bergisch Gladbach	02204/ 65254 02204/ 65254	- -	- -	- -
	IBM, GENIE III, GENIE IIIs, SHARP Pocket 1600▪ Scanner								
3	Bernhardt	Helmut	Hafenstraße 7	GER 24262	Heikendorf	0431/ 241907 0431/ 77578-20	0431/ 245717 -	- 2:242/262.26	- -
	IBM, Prof 180, CPU 280▪ diverse PCs,Novellite-Netz,1496E,Mustek105+								
4	Bielenberg	Georg	Erikaweg 1	GER 24568	Kaltenkirchen	04191/ 3751 04193/ 90430	- -	- -	- -
	Atari 260ST, C128, Schneider Joyce ▪ Modem: LC2496 Digitech, Scanner: Supersc.III								
5	Brans	Jörg	Tieloh 55	GER 22307	Hamburg	040/ 6906531 -	- -	- -	- -
	IBM▪ CD-Rom, Streamer, Soundkarte								
6	Braun	Günter W.	Postfach 80 02 26	GER 81602	München	- -	- -	- -	- -
	Commodore 128 D▪ Akustikkoppler								
7	Braun	Harald	Postfach 8011	GER 24154	Kiel	0431/35139 -	- -	- -	- -
8	Böckling	Ulrich	Juchaczstraße 61	GER 56203	Höhr-Grenzhausen	02624/ 4861 02631/ 895168	- -	- -	- -
	IBM 386DX33,1040ST,VC20,C64,TRS80 M I,ZX81 ▪ Modem, Videodat-, Videotextdecoder								
9	Chwolka	Fritz	Saarstraße 34	GER 52457	Aldenhoven	02464/ 8920 -	- -	- 2:248/242:8	- -
	IBM 386+Co, Commodore, Apple, Z 280▪ Modem 2.4								
10	Dose	Volker	Dorfstraße 10	GER 24235	Brodersdorf	04343/ 1357 -	- -	- -	- -
	GENIE IIIs mit Z180▪ EPROMer, Modem								
11	Halgasch	Gert	Großschönauer Straße 26	GER 02796	Jonsdorf	035844/ 636 -	- -	- -	- -
	IBM 386DX▪								
12	Hartmann	Hans-Günther	Möwenstraße 9	GER 27804	Berne	04406/ 6911 0421/ 248-2419	04406/ 1071 -	- 2:240/300.24	- -
	IBM 386SX, TANDY M4p, Z280-Kartenrechner▪ Real Time Clock, Speed Up Kit 6,3MHz, 40MB-P1								
13	Hebecker	Ulrich	Büsnauer Straße 15	GER 70563	Stuttgart	0711/ 734800 -	- -	- -	- -
	IBM 286+386, 128D, 1581, Kaypro484▪								
14	Held	Manfred	Stirner Straße 22	GER 91785	Pleinfeld	09144/ 6563 0911/ 219-2245	09144/ 8514 -	- 2:2400/10.10	- -
	IBM▪ Modem ZyXEL 1496+, CD-ROM								
15	Hermann	Klaus	Forchenstraße 8	GER 72124	Pliezhausen	07127/ 71945 -	- -	- 2:2407/70.740	Discovery 07127/70107
	IBM 386▪ ET 4000, Modem								
16	Hürdler	Manfred	Niederhoferstraße 29	GER 97222	Rimpar	09365/ 4235 -	- -	- -	- -
	Victor Sirius 1 (IBM), CPC 6128▪								
17	Johnen	Willi	Hansemannstraße 1	GER 52351	Düren	02421/ 501305 02421/ 33064	- -	- -	- -
	GENIE IIIs▪								
18	Kauka	Dietmar	Straße des Friedens 37	GER 04552	Neukirchen (Borna/Leipz	03433/851019 -	- -	- -	- -
19	Kemmer	Jürgen	Dorfberg 7	GER 97232	Sulzdorf	09334/ 1050 -	- -	- 2:247/2086.1	- -
	IBM 386, UltraSound, ZyXEL U1496E+▪ DCF-77-Empfänger parallel, IR-Sender seriell								
20	Kuhn	Eckehard	Im Dorf 14	GER 72636	Frickenhausen	07022/ 45417 -	- -	- -	- -
	Atari ST 1040, TRS80 M I▪								
21	Linder	Jörg	Küstriner Str. 68	GER 15306	Seelow	03346/ 520 -	- -	- -	- -
	KC 85/4 mit Floppy▪								
22	Lorenz	Walter	Mahräckerstraße 9	GER 60431	Frankfurt /Main	069/ 531656 -	- -	- -	- -
	IBM 286/486, Z80-, HD 64180-Eigebau, Z280-T.R▪ Scanner, Soundkarte, ET 4000								
23	Magnus	Andreas	Bismarckstraße 29	GER 45879	Gelsenkirchen	0209/ 144029 -	- -	- -	- -
	IBM 386Dx, GENIE IIIs▪ Modem 2400								
24	Mahlert	Herbert	Hohenbudbergerstraße 112 A	GER 47229	Duisburg	02065/ 47217 02065/ 902592	- -	- -	- -
	IBM, GENIE I▪ c't Videotext-Karte, Vobis Videodat-Decoder								
25	Müller	Kurt	Sophie-Scholl-Ring 3B	GER 21502	Geesthacht	04152/ 70643 040/ 89983403	- -	- -	- -
	IBM, Atari Mega ST4 ▪ ATONCE 386,OverScan,Scanner,HBS640-T36,Modem								