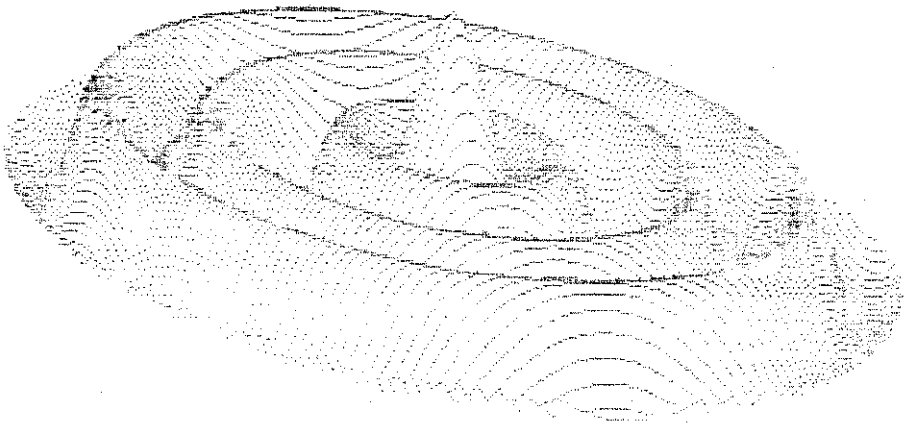
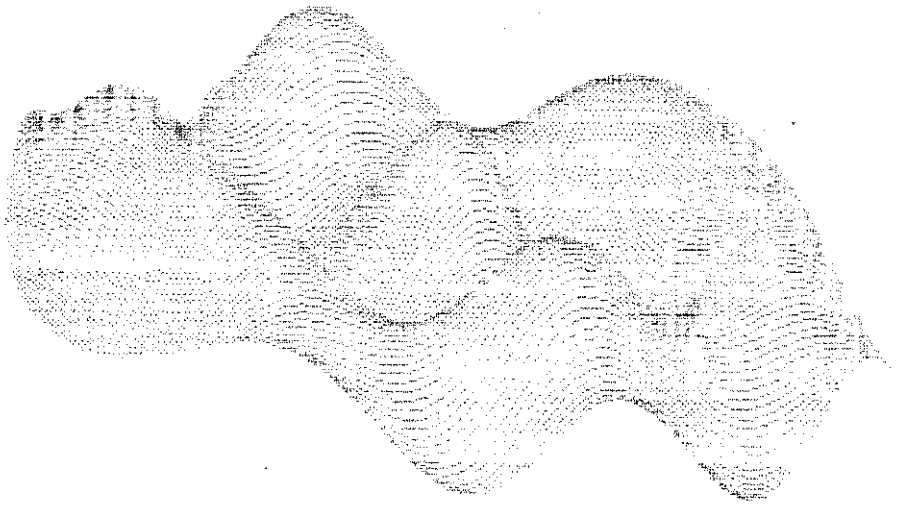
```
10  REM ..................................................
15    REM ................... ZYLINDER ....................
20    REM ..................................................
25    REM ............................... (c) Rolf Schall 1984.
30    REM ..................................................
35    REM ... Sehr schoenes Demonstrationsprogramm zur .....
40    REM ... Erzeugung von Mischfarben in MODE 7. Durch ...
45    REM ... geeignete Kombinationen werden weiche Farb- ..
50    REM ... uebergaenge erzielt, was den Eindruck glaen-..
55    REM ... zender Metallzylinder hervorruft. ............
60    REM ... Auf RGB-Monitoren koennen die Farbwerte ......
65    REM ... abweichen. Dort muessen dann andere Kombina-..
70    REM ... tionen benutzt werden (ab Zeile 9010). Die ...
75    REM ... Farben wurden mit dem Hilfsprogramm PALETTE ..
80    REM ... ausgewaehlt. ................. Viel Spass! ....
85    REM ..................................................
90    *** IMP INT
100   CLEAR 1000:SUPERRES=#300:GOSUB 700
110   COLORG 8 0 0 0:CALLM SUPERRES
120   DIM BA(7,5),FO(7,5)
150   FOR I=0 TO 5:FOR J=0 TO 7:READ BA(J,I),FO(J,I):NEXT:NEXT
210   R=RND(XMAX/8-16):X=R*8:COL=RND(6)
220   Y1=RND(YMAX):Y2=RND(YMAX):GOSUB 500
230   H=RND(5)+1:R=RND(YMAX/H-16):Y=R*H:COL=RND(6)
240   X1=INT(RND(XMAX/8))*8:X2=INT(RND(XMAX/8))*8-1
250   IF X1>X2 THEN 240:GOSUB 600:GOTO 210
500   REM ........ VERTIKALE ZYLINDER .....................
510   X1=X:X2=X+120:FOR I=0 TO 7:BA=BA(I,COL)
520   FILL X1,Y1 X1+7,Y2 BA:FILL X2,Y1 X2+7,Y2 BA
530   X1=X1+8:X2=X2-8:NEXT
540   X1=X:X2=X+120:FOR I=0 TO 7:FO=FO(I,COL)
545   IF FO=BA(I,COL) THEN 570
550   FOR J=0 TO 8 STEP 2:X3=X1+J:X4=X2+J
560   DRAW X3,Y1 X3,Y2 FO:DRAW X4,Y1 X4,Y2 FO:NEXT
570   X1=X1+8:X2=X2-8:NEXT
580   RETURN
600   REM ...... HORIZONTALE ZYLINDER .....................
610   H1=H-1:Y1=Y:Y2=Y+15*H:FOR I=0 TO 7:BA=BA(I,COL)
620   FILL X1,Y1 X2,Y1+H1 BA:FILL X1,Y2 X2,Y2+H1 BA
630   Y1=Y1+H:Y2=Y2-H:NEXT
640   Y1=Y:Y2=Y+15*H:FOR I=0 TO 7:FO=FO(I,COL)
645   IF FO=BA(I,COL) THEN 670
650   FOR J=X1 TO X2 STEP 2
660   DRAW J,Y1 J,Y1+H1 FO:DRAW J,Y2 J,Y2+H1 FO:NEXT
670   Y1=Y1+H:Y2=Y2-H:NEXT
680   RETURN
700   START=SUPERRES:ADRES=START
710   READ BYTE:IF BYTE<#100 THEN POKE ADRES,BYTE:ADRES=ADRES+1:GOTO 710
720   IF BYTE=#FFF THEN RETURN
730   READ LBYTE,HBYTE:POKE ADRES,BYTE IAND #FF
740   LBYTE=LBYTE+(START IAND #FF):CARRY=LBYTE SHR 8
750   POKE ADRES+1,LBYTE IAND #FF:HBYTE=HBYTE+(START SHR 8)+CARRY
760   POKE ADRES+2,HBYTE:ADRES=ADRES+3:GOTO 710
```

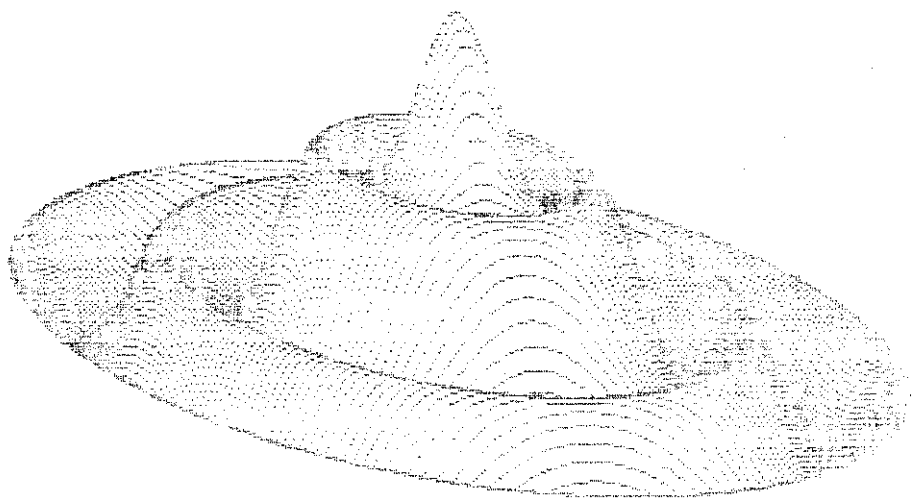
```
770   DATA #F5,#C5,#D5,#E5,#3E,#8,#EF,#18,#2A,#9E,#0,#3E,#F,#A5,#87,#87
780   DATA #87,#87,#6F,#7C,#E6,#F,#85,#E,#FF,#21,#EF,#BF,#1E,#F4,#16,#42
790   DATA #36,#B0,#2B,#36,#40,#2B,#71,#2B,#77,#15,#1C2,#25,#0,#2B,#1D,#1C2
800   DATA #1E,#0,#121,#6B,#0,#11,#84,#0,#6,#15,#7E,#12,#13,#23,#5,#1C2
810   DATA #3A,#0,#21,#27,#40,#22,#A5,#2,#6,#10,#21,#F0,#BF,#11,#28,#40
820   DATA #7E,#12,#13,#23,#5,#1C2,#50,#0,#3E,#BF,#32,#2B,#40,#32,#2F,#40
830   DATA #32,#33,#40,#32,#37,#40,#E1,#D1,#C1,#F1,#C9,#27,#40,#77,#B0,#37
840   DATA #40,#27,#40,#37,#40,#27,#40,#77,#56,#EF,#56,#0,#2,#F4,#2C,#86
850   DATA #FFF
900   REM ......... Vorder- und Hintergrundfarben ..........
910   DATA 0,5,3,5,4,5,5,6,5,5,5,13,5,15,13,15
920   DATA 0,1,1,1,1,9,1,12,9,9,9,12,12,12,12,15
930   DATA 0,3,3,4,3,6,3,3,3,10,3,15,10,10,10,15
940   DATA 0,0,0,7,0,8,0,15,8,8,7,12,8,15,15,15
950   DATA 1,7,4,6,6,6,6,7,7,7,3,14,7,14,8,14
960   DATA 0,2,2,2,2,8,2,11,2,15,4,15,11,11,11,15
```





```
10    REM ..................................................
15    REM ................. FARB-PALETTE ..................
20    REM ............. (c) Rolf Schall 1984 ..............
25    REM ..................................................
30    REM Im MODE 7 laesst sich mit  einem Kniff die beacht-
35    REM liche Zahl von 128 Farben produzieren, indem dicht
40    REM nebeneinander farbige Punkte gesetzt werden. .....
45    REM Durch die niedrige Aufloesung des Monitors entste-
50    REM hen neue Farben, die freilich von Geraet zu Geraet
55    REM unterschiedlich ausfallen koennen. Dieses Programm
60    REM erlaubt es, aus den 128 Moeglichkeiten eigene Kom-
65    REM binationen auszuwaehlen. Hierzu die beiden Cursors
70    REM an die entsprechenden Grundfarben  setzen und eine
75    REM Ziffer von 1 bis 8 eingeben. Abbruch mit SPACE ...
80    REM ..................................................
90    REM IMP INT
100   SUPERRES=#300:GOSUB 500
110   COLORG 8 0 0 0:CALLM SUPERRES
120   SY=14:SX=24:XM=XMAX-80:X=48:YM=YMAX-18:Y=YM
130   FOR COL=0 TO 15
140   FILL X,Y-SY XM,Y COL:FILL XM+8,Y-SY+1 XM+20,Y-1 COL
150   Y=Y-SY:X=X+SX:NEXT
160   X=48:Y=YM-SY
170   FOR COL=0 TO 15:FILL X+2,YM+2 X+SX-2,YMAX-8 COL
180   FOR X2=X TO X+SX STEP 2:DRAW X2,Y X2,YM COL
190   NEXT:X=X+SX:Y=Y-SY:NEXT
200   REM ..................... ZIFFERN MALEN ..................
210   X=48:SX2=32
220   FOR I=1 TO 8:READ NUM,LINES:FOR J=1 TO LINES
230   READ X1,Y1,X2,Y2:DRAW X+X1,Y1 X+X2,Y2 0:NEXT
240   X=X+SX2:NEXT
300   REM ................... CURSORSTEUERUNG .................
310   G=GETC:IF G<16 THEN 310:IF G>56 THEN 310
320   ON G-15 GOTO 340,350,370,380
330   IF G=32 THEN MODE 0:END
335   IF G<49 THEN 310:IF FO<BA THEN 310:GOSUB 400:GOTO 310
340   BA=BA+14
350   BA=(BA+1) MOD 16:Y=YM-BA*SY-SY
360   FILL XM+24,YO XM+32,YO+SY 8:FILL XM+24,Y XM+32,Y+SY 0
365   YO=Y:GOTO 310
370   FO=FO+14
380   FO=(FO+1) MOD 16:X=48+FO*SX
390   FILL XO,YMAX-6 XO+SX,YMAX 8:FILL X,YMAX-6 X+SX,YMAX 0
395   XO=X:GOTO 310
400   REM ............ Einzelne Mischfarben malen ............
410   MIX=(G-49)*SX2+48:FILL MIX,SY MIX+SX,SY+SY BA
420   FOR I=MIX TO MIX+SX STEP 2:DRAW I,SY I,SY+SY FO:NEXT
430   RETURN
500   REM ................. MLP fuer MODE 7 ..................
510   START=SUPERRES:ADRES=START
520   READ BYTE:IF BYTE<#100 THEN POKE ADRES,BYTE:ADRES=ADRES+1:GOTO 520
530   IF BYTE=#FFF THEN RETURN
540   READ LBYTE,HBYTE:POKE ADRES,BYTE IAND #FF
550   LBYTE=LBYTE+(START IAND #FF):CARRY=LBYTE SHR 8
560   POKE ADRES+1,LBYTE IAND #FF:HBYTE=HBYTE+(START SHR 8)+CARRY
570   POKE ADRES+2,HBYTE:ADRES=ADRES+3:GOTO 520
580   DATA #F5,#C5,#D5,#E5,#3E,#8,#EF,#18,#2A,#9E,#0,#3E,#F,#A5,#87,#87
590   DATA #87,#87,#6F,#7C,#E6,#F,#85,#E,#FF,#21,#EF,#BF,#1E,#F4,#16,#42
600   DATA #36,#B0,#2B,#36,#40,#2B,#71,#2B,#77,#15,#1C2,#25,#0,#2B,#1D,#1C2
610   DATA #1E,#0,#121,#6B,#0,#11,#84,#0,#6,#15,#7E,#12,#13,#23,#5,#1C2
```

```
›20    DATA #3A,#0,#21,#27,#40,#22,#A5,#2,#6,#10,#21,#F0,#BF,#11,#28,#40
›30    DATA #7E,#12,#13,#23,#5,#1C2,#50,#0,#3E,#BF,#32,#2B,#40,#32,#2F,#40
›40    DATA #32,#33,#40,#32,#37,#40,#E1,#D1,#C1,#F1,#C9,#27,#40,#77,#B0,#37
›50    DATA #40,#27,#40,#37,#40,#27,#40,#77,#56,#EF,#56,#0,#2,#F4,#2C,#86
›60    DATA #FFF
'00    REM ..............  ZIFFERN-DATEI ......................
'10    DATA 1,3,1,0,3,0,2,0,2,6,1,5,1,5
'20    DATA 2,5,0,0,4,0,1,1,3,3,4,4,4,5,1,6,3,6,0,5,0,5
'30    DATA 3,6,0,1,0,1,1,0,3,0,4,1,4,2,3,3,2,4,3,5,3,5,0,6,4,6
'40    DATA 4,3,0,3,3,6,0,2,4,2,3,0,3,4
'50    DATA 5,6,0,1,0,1,1,0,3,0,4,1,4,3,0,4,3,4,0,5,0,6,1,6,4,6
'60    DATA 6,5,1,0,3,0,0,1,0,3,4,1,4,2,2,3,3,3,1,4,3,6
'70    DATA 7,3,1,0,1,2,2,3,4,5,0,6,4,6
'80    DATA 8,7,1,0,3,0,1,3,3,3,1,6,3,6,0,1,0,2,0,4,0,5,4,1,4,2,4,4,4,5
```



GP 17,2

```
10

20    REM ............... The Wall ...........................
30    REM ..................................................
40    REM ............................. (c) 1984 by R.Schall.
50    REM ..................................................
60    REM Ein weiteres Grafikdemo  aus der Schall-Serie. Zu-
70    REM erst wird  eine Mauer  nach teilweise  zufaelligem
80    REM Muster aufgebaut.  Im zweiten  Schritt werden ein-
90    REM zelne Steine  herausgenommen und die Mauer dadurch
100   REM schrittweise  demontiert.  Wenn zwei nebeneinander
110   REM liegende Steine fehlen,  faellt der darueber haen-
120   REM gende  herunter.  Mit  diesem simplen  Algorithmus
130   REM fallen schon einige Steine,  aber noch lange nicht
140   REM alle.  Wer weiss einen ausgefeilteren Algorithmus,
150   REM der alle  Faelle  beruecksichtigt  (dieser braucht
160   REM nicht allzu schnell zu sein)?  Dasselbe  Programm
170   REM habe ich  gleich auch  noch in PASCAL uebertragen,
180   REM um die Geschwindigkeits- und  Strukturunterschiede
190   REM beider Sprachen einmal deutlich zu machen. .......
195   REM IMP INT
200   REM ................... Aufbau ...........................
210   COLORG 3 5 3 15:MODE 6
220   FOR I=0 TO YMAX-12 STEP 12:DRAW 0,I XMAX,I 15
230   OFFSET=15-OFFSET
240   FOR J=OFFSET+5 TO XMAX-15 STEP 30:X=J+RND(10)
250   FILL X-1,I+1 X,I+11 15:NEXT J:NEXT I
260   FILL 0,I XMAX,I 15:FILL 0,0 2,I 15
270   FILL 0,0 XMAX,1 15:FILL XMAX-2,0 XMAX,I 15
280   X=RND(XMAX-10)+5:Y=INT(RND(20))*12+1
290   IF SCRN(X,Y)=15 THEN 280:GOSUB 400
300   FILL X0,Y X,Y+11 15
310   IF SCRN(X0-2,Y)=15 THEN X=X0:GOTO 330
320   IF SCRN(X+2,Y)<>15 THEN 280
330   Y=Y+12:IF SCRN(X,Y)=15 THEN 280:GOSUB 400
340   GOSUB 500
350   GOTO 280
400   REM ..... Suche Begrenzung
410   IF SCRN(X,Y)<>15 THEN X=X-1:GOTO 410
420   X0=X:X=X+1
430   IF SCRN(X,Y)<>15 THEN X=X+1:GOTO 430
440   RETURN
500   REM ..... Ziegelstein faellt
510   SX0=X0+1:SX2=X-1:SX1=(X0+X)/2:SY=Y-1
520   IF SCRN(SX0,SY-1)+SCRN(SX1,SY-1)+SCRN(SX2,SY-1)<>45 THEN 540
530   DRAW SX0,SY SX2,SY 3:DRAW SX0,SY+11 SX2,SY+11 15:SY=SY-1:GOTO 520
540   RETURN
```

```
! ..................................................... !
! ..............The Wall, PASCAL-Version ............. !
! ..................................................... !

VAR I,X0,X,Y,RNX,RNY,OFFSET,   !. Globale Variablen ..!
    PARX0,PARX,PARY: INTEGER;  !. Rueckgabeparameter .!

PROC MODE(M);
   BEGIN
   MEM[#7D00]:=M; CALL(#7FD)
   END;

PROC COLORG(C1,C2,C3,C4);
   BEGIN
   MEM[#7D00]:=C1; MEM[#7D01]:=C2; MEM[#7D02]:=C3;
   MEM[#7D03]:=C4; CALL(#82B)
   END;

PROC DRAW(X1,Y1,X2,Y2,C);
   BEGIN
   MEM[#7D03]:=X1 AND 255; MEM[#7D04]:=X1 SHR 8;
   MEM[#7D05]:=X2 AND 255; MEM[#7D06]:=X2 SHR 8;
   MEM[#7D01]:=Y1; MEM[#7D02]:=Y2; MEM[#7D00]:=C;
   CALL(#886)
   END;

PROC FILL(X1,Y1,X2,Y2,C);
   BEGIN
   MEM[#7D03]:=X1 AND 255; MEM[#7D04]:=X1 SHR 8;
   MEM[#7D05]:=X2 AND 255; MEM[#7D06]:=X2 SHR 8;
   MEM[#7D01]:=Y1; MEM[#7D02]:=Y2; MEM[#7D00]:=C;
   CALL(#8CC)
   END;

FUNC SCRN(X,Y);
   BEGIN
   MEM[#7D05]:=X AND 255; MEM[#7D06]:=X SHR 8;
   MEM[#7D02]:=Y; CALL(#912); SCRN:=MEM[#7D00]
   END;

FUNC XMAX;
   BEGIN
   MEM [#7D02]:=0;MEM [#7D05]:=0;MEM[#7D06]:=0;
   CALL(#912); XMAX:=256*MEM[#7D04]+MEM[#7D03]
   END;

FUNC YMAX;
   BEGIN
   MEM [#7D02]:=0;MEM [#7D05]:=0;MEM[#7D06]:=0;
   CALL(#912); YMAX:=MEM[#7D01]
   END;

FUNC RND(R); !.......... Hardware-Random ..............!
   VAR B,RN: INTEGER;  ! R: Random-Bereich !
   BEGIN
   B:=1; RN:=0;
   WHILE B<R DO
      BEGIN            ! Lies naechstes Random-Bit !
      B:=B+B; RN:=RN+RN+(MEM[#FD00] SHR 6);
      END;
   RN:=RN*R DIV B; IF RN<0 THEN RN:=-RN;
   RND:=RN;            ! Normieren !
   END;

PROC BORDER(X0,X,Y); !..... Suche Randbegrenzung ....!
   BEGIN
   WHILE SCRN(X,Y)<>15 DO
      X:=X-1;
   X0:=X; X:=X+1;
   WHILE SCRN(X,Y)<>15 DO
      X:=X+1;
   PARX0:=X0; PARX:=X;
   END;              ! X0,X,Y sind lokale Parameter!!!

PROC MOVE(X0,X,Y); !......... Stein faellt .........!
   VAR MX,Y1: INTEGER;
   BEGIN
   Y:=Y+12;
   IF SCRN(X,Y)<>15 THEN
      BEGIN
      BORDER(X0,X,Y);
      X0:=PARX0+1; X:=PARX-1;
      MX:=(X0+X) DIV 2;
      Y:=Y-1; Y1:=Y-1;
      WHILE SCRN(X0,Y1)+SCRN(MX,Y1)+SCRN(X,Y1)=45 DO
         BEGIN
         DRAW(X0,Y,X,Y,3); DRAW(X0,Y+11,X,Y+11,15);
         Y:=Y-1; Y1:=Y-1;
         END;
      END;
   END;

BEGIN ! ............... Hauptprozedur ............... !
COLORG(3,5,3,15); MODE(8);
Y:=0; OFFSET:=0;
WHILE Y<YMAX-12 DO
   BEGIN
   DRAW(0,Y,XMAX,Y,15);
   OFFSET:=15-OFFSET; X:=OFFSET+RND(10)+5;
   WHILE X<XMAX DO
      BEGIN
      FILL(X-1,Y+1,X,Y+11,15); X:=X+30;
      END;
   Y:=Y+12;
   END;
FILL(0,Y,XMAX,Y,15); FILL(0,0,2,Y,15);
FILL(0,0,XMAX,1,15); FILL(XMAX-2,0,XMAX,Y,15);

RNX:=(XMAX-30) DIV 15; RNY:=(YMAX-24) DIV 12;
WHILE 1 DO
   BEGIN
   REPEAT
      X:=RND(RNX)*15+15; Y:=RND(RNY)*12+14;
   UNTIL SCRN(X,Y)<>15;
   BORDER(X0,X,Y);
   X0:=PARX0; X:=PARX;
   FILL(X0,Y-1,X,Y+10,15);
   IF SCRN(X0-2,Y)=15 THEN MOVE(X0,X0,Y);
   IF SCRN(X+2,Y)=15 THEN MOVE(X,X,Y);
   END;
END.
```

Spiegelungen

Ich habe neulich experimentiert mit der Erzeugung von Bildern, die eine Art Höhenliniendarstellung einer mathematischen Funktion wiedergeben--dazu wird für jeden der 80.000 Mode-6 Bildschirmpunkten ein Funktionswert berechnet und der Punkt entsprechend diesem Wert eingefärbt. Das verlangt natürlich sehr viel Rechenzeit (bei einem typischen Bild über eine Stunde!). Wenn aber die Funktionen und die Bilder, wie oft der Fall ist, Symmetrien aufweisen, dann braucht man nur einen entsprechenden Teil des Bildes auszurechnen, da man aus diesem Teil durch Spiegelung das ganze Bild gewinnen kann.

Ich möchte in diesem Bericht einige Assemblerprogramme vorstellen, die solche Spiegelungen sehr schnell (schlimmstenfalls 4,2 Sekunden) ausführen und so zu einem erheblichen Zeitgewinn bei der Erstellung von Bildern beitragen können. Dabei werden fünf Spiegelungsarten berücksichtigt: nicht nur Spiegelung in den waagerechten und senkrechten Koordinatenachsen, sondern auch in den Diagonalen (also in den Geraden x=y und x=-y), und schließlich die radiale Spiegelung im Bildschirmmittelpunkt. Ferner können bei all diesen Spiegelungsarten nicht nur symmetrische Bilder erstellt werden, sondern auch antisymmetrische, bei denen im Spiegelbild die Farben umgekehrt werden (dies entspricht dem Fall, daß die dargestellte mathematische Funktion im Spiegelpunkt nicht den gleichen, sondern den negativen Wert hat zum Wert im ursprünglichen Punkt). *Umkehrung der Farben* bedeutet, daß die Farben 20 und 23 vertauscht werden, und die Farben 21 und 24 vertauscht werden.

Man beachte, daß die Spiegelungsprogramme nicht den Bildschirm als **ganzes** spiegeln, sondern immer nur eine Hälfte des Bildschirms spiegelbildlich in die andere Hälfte kopieren (wobei diese überschrieben, also zerstört wird!)

Die Unterprogramme werden von Basic aus aufgerufen, wobei eine Variable übergeben werden muß, die angibt, ob symmetrisch oder antisymmetrisch gespiegelt werden soll. Hat diese Variable den Wert 0, so wird keine Spiegelung ausgeführt. Dies erleichtert die Anwendung dieser Programme etwas, denn schon bei der Berechnung der Variablen kann entschieden werden, ob die entsprechende Spiegelung überhaupt auszuführen ist. Es ist dann später keine zusätzliche IF ... THEN CALLM ... Abfrage dafür nötig.

Als Beispiel ist ein Basicprogramm aufgeführt, das einige Hilfen bietet bei der Verwendung der Spiegelungsprogrammen. Ein Bild kann nämlich mehrere Symmetrien gleichzeitig aufweisen, und aus dem Vorhandensein einiger Symmetrien kann die Existenz anderer Symmetrien automatisch folgen (z.B., ein Bild, das in einer Koordinatenachse und in einer Diagonalen symmetrisch ist, ist immer auch in der anderen Achse, der anderen Diagonalen und im Bildmittelpunkt symmetrisch). Damit man nicht bei jedem Bild lange überlegen muß, welche Symmetriearten relevant sind, in welchem Grundbereich man das Bild zuerst erstellen muß, und wie man zum Schluß spiegeln muß, stellt das Basicprogramm durch eine effiziente und möglichst kurze Folge von Fragen an den Benutzer fest, welche Symmetriearten vorhanden sind, berechnet die Grenzen des Grundbereichs, und bestimmt, welche der Spiegelungsprogramme aufzurufen sind, und mit welchem Wert der zu übergebenden Variablen, um aus dem Teilbild im Grundbereich das ganze Bild zu erzeugen. Dabei wird, wie oben vorgeschlagen, für die nicht auszuführenden Spiegelungsprogrammen der Variablenwert 0 festgelegt. Später werden ohne Abfrage alle Spiegelungsprogramme aufgerufen, aber nur die gewünschten Spiegelungen werden tatsächlich ausgeführt.

Dies nur zur Einführung. Alles weitere, sowie detaillierte Angaben über die Verwendung der Spiegelungsprogrammen sowie ihrer genauen Funktionsweise, finden Sie in folgendem gut kommentiertem Listing:

```
          TITL            Spiegelung des Bildschirms
          REM
          ORG    :300
          MEM    :5300
EXIT      EQU    :C14D    POP alle Register und RET
LINLEN    EQU    90       Bytelänge einer Mode 6 Bildschirmzeile
          REM
* Die Einsprünge XREF,YREF,CREF,ANTIREF,EQREF (ab der Adresse
* #300) werden von Basic aufgerufen mit CALLM PROG,FLAG%.  Wenn
* FLAG%=0, geschieht nichts.  Bei FLAG% ≠ 0 wird ein Teil des
* Bildschirms spiegelbildlich auf die andere Hälfte des Bild-
* schirms kopiert (wobei die Farben umgekehrt werden, wenn
* FLAG%<0).  Folgende Tabelle gibt für die fünf Unterprogramme
* an, wie gespiegelt wird, aus welchem Teil des Bildschirms das
* zu spiegelnde Bild entnommen wird, und in welchen Bereich hin-
* ein das gespiegelte Bild kopiert wird.  Dabei beziehen sich
* die Koordinaten x und y auf den Bildschirmmittelpunkt als
* Koordinatenursprung (x = y = 0).
*
* Unter-     spiegelt     Quellbereich          Zielbereich
* programm     in
*
* XREF       y-Achse      rechts der Achse      links der Achse
*
* YREF       x-Achse      oberhalb              unterhalb
*
* CREF       Ursprung     rechts oberhalb       links unterhalb
*            (radial)     der Geraden x=-y      der Geraden x=-y
*                         (auch Gerade selber   (auch Gerade selber
*                         für x>0)               für x<0)
*
* ANTIREF  Gerade x=-y    rechts oberhalb       links unterhalb
*
* EQREF    Gerade x=y     rechts unterhalb      links oberhalb
*
* Alle Programme lassen den Quellbereich unverändert und über-
* schreiben den Zielbereich!
* Alle Programme erhalten den BC-Register und zerstören die
* anderen Register.
          REM
          JMP    XREF
          JMP    YREF
          JMP    CREF
          JMP    ANTIREF
* EQREF spiegelt in der Geraden x=y
EQREF     CALL   GETFLG   hole Wert von FLAG% (Farbumkehrmarke)
          LXI    H,:6645-10+(7*90)  erster 8x8 Bit Quellblock
          LXI    D,8*90   Adressunterschied zwischen Zielblöcken
          REM
          REM        Folgendes DIAGRF spiegelt Bild in beliebiger
          REM        Diagonale, in welcher hängt von Vorzeichen
          REM        von DE ab.  Der Quellbereich wird streifen-
          REM        weise verarbeitet, und jeder Streifen (8
          REM        Zeilen tief) wird von links nach rechts in
          REM        8-Bit breite Blöcke zerlegt, die in den
          REM        Hilfsspeicher TAB kopiert und dort gespiegelt
          REM        werden (Unterprogramm CONVERT).
          REM
          REM
```

```
DIAGRF    PUSH  H         merke Adresse des ersten Blocks
          CALL  EQRF5     spiegele höherwertigen Bildbits
          POP   H         und dann
          DCX   H         niederwertigen Bildbits
          REM             (Beachte jeder Bildschirmpunkt wird durch
          REM             zwei Bits beschrieben!!)
EQRF5     PUSH  B         Hier wird nur einer der zwei Bits
          REM             der Bildschirmpunkte gespiegelt, also
          REM             nur jeder zweite Byte wird verarbeitet
          MVI   B,32      soviele 8-Zeilen tiefe Bildstreifen zu
          REM             verarbeiten
EQRF10    PUSH  H         merke Quelladresse
          PUSH  H         Quelladresse ist erste Zieladresse
          MOV   C,B       soviele 8-Punkt breite Blöcke in diesem
          REM             Bildstreifen
EQRF20    MOV   A,D       negativ für Spiegelung in x=-y!!
          ADD   A         setze Carrybit entsprechend!
          LDA   FLAG      Farbumkehrmarke
          CALL  SCRTAB    kopiere Block zu Hilfstabelle
          CALL  CONVERT   spiegele Hilfstabelle diagonal
          DCX   H
          DCX   H         zeige auf nächsten Quellblock
          XTHL            merke Quellzeiger, hole Zielblockadresse
          MOV   A,C       wenn nicht auf Bilddiagonale, also
          CMP   B         wenn maskieren nicht nötig, setze Carry
          REM             (Blöcke auf Diagonale werden maskiert
          REM             damit Quellbereich bei Spiegelung nicht
          REM             überschrieben wird).
          MOV   A,D       negativ für Spiegelung in x=-y!!
          RAL             setze Carry entsprechend und
          REM             Bit 0 für maskiere
          CALL  TABSCR    kopiere Spiegelbild zurück zu Bildschirm
          DAD   D         nächste Zielblockadresse!
          XTHL            merke Ziel-, hole Quellblockadresse
          DCR   C         zähle Blöcke, dieser Streifen fertig?
          JNZ   EQRF20    nein, nächster Block
          POP   H         dieser Streifen fertig, vergesse Ziel
          POP   H         hole Anfangsadresse dieses Streifens
          DCX   H         nächster Streifen beginnt 8 Punkte
          DCX   H         nach rechts!
          DAD   D         Anfangsadresse des nächsten Streifens
          DCR   B         zähle Streifen, alle fertig?
          JNZ   EQRF10    nein, nächster Streifen
          POP   B         stelle BC wieder her und
          RET             fertig
* ANTIREF spiegelt in der Geraden x=-y
ANTIREF   CALL  GETFLG    hole Wert von FLAG% (Farbumkehrmarke)
          LXI   H,:BFEB-10  erster 8x8 Bit Quellblock
          LXI   D,-8*90   Adressunterschied zwischen Zielblöcken
          JMP   DIAGRF    führe Spiegelung aus
* CREF spiegelt radial im Bildschirmmittelpunkt
* Das Bild wird Zeile um Zeile verarbeitet, wobei die Quell-
* zeilen von rechts nach links, beginnend bei der Diagonalen,
* abgearbeitet werden.  Die Reihenfolge der Bits wird umgekehrt
* und das Ergebnis in den Zielbereich geschrieben. Die Zielzeilen
* werden von der Diagonalen nach rechts geschrieben.  Es werden
* zuerst nur die höherwertigen Bildbits gespiegelt, anschließend
* dann alle niederwertigen Bildbits.
CREF      CALL  GETFLG    hole Wert von FLAG% (Farbumkehrmarke)
          LXI   H,:65F3+10  Anfang Quellbereich höherwertige Bits
          LXI   D,:BFEB-10  Anfang Zielbereich höherwertige Bits
          CALL  CRF1      spiegele höherwertige Bildbits
```

```
        LXI   H,:65F2+10  Anfang Quellbereich niederwertige Bits
        LXI   D,:BFEA-10  Anfang Zielbereich niederwertige Bits
        REM               Folgender Teil spiegelt nur ein Bit
        REM               jedes Bildschirmpunktes!
CRF1    PUSH  B
        MVI   B,6         Länge in 8-Bit Blöcke der kürzesten
        REM               zu verarbeitenden Bildschirmzeile
CRF3    MVI   A,:7F       Maske für Bildschirmdiagonale, damit
        REM               Spiegelbild Quellbild nicht überschreibt
CRF5    PUSH  B           merke Zeilenlänge
        PUSH  H           merke Quelladresse der Zeile
        PUSH  D           merke Zieladresse der Zeile
        PUSH  D           Zieladresse auf Stapel, gebe DE frei.
        MOV   D,A         Maske für Diagonalblock in dieser Zeile
CRF10   MOV   E,A         Maske für jetzigen 8-Bit Block
        LDA   FLAG        verwandle Vorzeichen von FLAG
        ADD   A           in eine
        SBB   A           Maske für die
        XRA   M           eventuell verlangte Farbumkehrung
        CALL  INVERT      kehre Reihenfolge der Bits in A um
        ORA   E           und lösche im Ergebnis Bits die
        XRA   E           dem Quellbereich entsprechen!
        MOV   C,A         rette Spiegelbildbyte nach C
        DCX   H           Quellzeiger auf nächsten
        DCX   H           Quellblock
        XTHL              hole Zieladresse, merke Quelladresse
        MOV   A,M         hole Zielbyte aber
        ANA   E           behalte Bits aus dem Quellbereich!
        ORA   C           und schreibe die neuen Zielbits
        MOV   M,A         speichere wieder im Bildschirm ab
        INX   H           nächste
        INX   H           Zielblockadresse
        XTHL              vertausche Ziel-, Quelladresse
        XRA   A           löscht Diagonalmaske für alle Folge-
        REM               blöcke einer Zeile! (also wirklich
        REM               maskiert wird nur der erste Block der
        REM               Zeile, nur der liegt auf der Diagonalen!
        DCR   B           zähle Blöcke in Zeile
        JNZ   CRF10       mache nächsten Block wenn nicht fertig
        MOV   A,D         Diagonalmaske nach A
        POP   H           vergesse letzte Quell-, Zieladressen
        POP   H           erste Zieladresse der letzten Zeile
        LXI   D,-90       ergibt erste Zieladresse der nächsten
        DAD   D           Zeile
        XTHL              Quelladresse der alten Zeile
        LXI   D,90        berechne Quelladresse der nächsten
        DAD   D           Zeile, beachte Carry jetzt gelöscht!!!
        POP   D           Zieladresse nach DE
        POP   B           hole zu verarbeitende Zeilenlänge
        RAR               verschiebe Diagonalmaske
        REM               (Diagonalpunkt der nächsten Zeile·
        REM               ist ja ein Bit versetzt)
        JC    CRF5        bei Carry, Maske noch nicht verbraucht
        REM               dies bedeutet: nächste Bildzeile berührt
        REM               gleiche Anzahl zu verarbeitender Bytes.
        REM               Aber alle 8 Zeilen hat Diagonalpunkt
        REM               sich um einen ganzen Byte verschoben,
        DCX   D           also Ziel beginnt einen Bildbyte nach
        DCX   D           rechts,
        INX   H           Quelle beginnt einen Bildbyte weiter
        INX   H           links und in jeder Zeile ist
        INR   B           ein Byte mehr zu verarbeiten
```

```
          MVI   A,38      maximale Zeilenlänge (zu verarbeitende
          CMP   B         Bytes) ist 37, also
          JNZ   CRF3      bei B = 38 ist Schluß
          POP   B         BC wiederherstellen
          RET             und fertig
* XREF spiegelt in der y-Achse.  Verarbeitung geschieht
* zeilenweise, beginnend mit der obersten Bildschirmzeile,
* und in jeder Zeile beginnend an der Zeilenmitte. Die Bytes der
* rechten Bildschirmhälfte werden durch Umkehrung der Bitreihen-
* folge gespiegelt und von der Mitte nach links in die linke
* Bildhälfte geschrieben.
XREF      CALL  GETFLG    hole Wert von FLAG% (Farbumkehrmarke)
          LXI   H,:BFEB-42  erste Quelladresse
          REM               (mitte der obersten Zeile)
          PUSH  B
          LDA   FLAG      berechne aus FLAG
          ADD   A         eine Maske
          SBB   A         (0 oder #FF) für die eventuell
          MOV   C,A       gewünschte Farbumkehr
          REM             funktioniert auch wenn FLAG negativ ≠ -1
          XRA   A         Zeilenzähler, verarbeite 256 Zeilen
XRF5      MOV   D,H       Quelladresse nach DE, gleiche Adresse
          MOV   E,L       zeigt direkt unter Zielbereich, belasse
          REM             also auch als Zielzeiger in HL
          PUSH  PSW       Zeilenzähler
          MVI   B,21      soviele 8-Punkt Blöcke in jeder
          REM             Zeile zu verarbeiten
XRF10     CALL  INV00     hole höherwertige Quellbits und spiegele
          INX   H         und speichere als
          INX   H         höherwertige Zielbits
          MOV   M,A       ab
          CALL  INV00     hole niederwertige Quellbits, spiegele
          DCX   H         speichere als niederwertige
          MOV   M,A       Zielbits ab
          INX   H         aktualiziere Zielzeiger
          DCR   B         zähle 8-Punkt Blöcke in der Zeile
          JNZ   XRF10     wenn nicht fertig, mache nächsten Block
          LXI   D,-90-42  Abstand jetziger Quellzeiger zu
          REM             Anfang Quellbereich in nächster Zeile
          DAD   D         Quellanfang nächste Zeile in HL
          POP   PSW       zähle
          DCR   A         Zeilen
          JNZ   XRF5      nicht fertig? dann weiter
          POP   B         BC wiederherstellen
          RET             und fertig
* YREF spiegelt in der x-Achse.
* Beginnend an der Bildschirmmitte werden die Zeilen der oberen
* Bildschirmhälfte (von unten nach oben gelesen) von oben nach
* unten in die untere Bildschirmhälfte kopiert.
YREF      CALL  GETFLG    hole Wert von FLAG% (Farbumkehrmarke)
          LXI   H,:BFEB-(127*90)  Anfang Quellbereich
          LXI   D,:BFEB-(128*90)  Anfang Zielbereich
          PUSH  B
          LDA   FLAG      berechne aus FLAG
          ADD   A         eine Maske
          SBB   A         (0 oder #FF) für die eventuell
          MOV   C,A       gewünschte Farbumkehr
          REM             funktioniert auch wenn FLAG negativ ≠ -1
          XRA   A         Zeilenzähler, verarbeite 128 Zeilen!
          PUSH  PSW       (zählt aufwärts, zählt aus bei :80!)
          MVI   B,84      zu verarbeitende Byteanzahl pro Zeile
```

```
YRF10    MOV   A,M       hole Quellbyte
         XRA   C         eventuelle Farbumkehr
         STAX  D         speichere in Zielbereich
         DCX   H         Speicherzeiger
         DCX   D         aktualisieren
         DCR   B         und Byte in Zeile zählen
         JNZ   YRF10     wenn nicht fertig, nächstes Byte machen
         MOV   A,C       Farbumkehrmaske nach A, mache BC frei!
         LXI   B,-6      nach Ende der Zeile beginnt nächste
         XCHG            Zielzeile (eine Zeile tiefer als die
         DAD   B         jetzige) 6 Byte unter letzten
         XCHG            Zielzeigerwert
         LXI   B,-6+90+90 und nächste Quellzeile ist eine höher
         DAD   B         als die letzte
         MOV   C,A       Farbumkehrmaske zurück nach C
         POP   PSW       zähle
         INR   A         128 Zeilen
         JP    YRF5      (solange positiv, noch nicht fertig)
         POP   B         BC wiederherstellen
         RET             und fertig
* GETFLG speichert den Wert der vom CALLM übergebenen Basic-
* variablen an der Adresse FLAG ab. Wenn dieser Wert 0 war, er-
* folgt ein Abbruch der weiteren Verarbeitung durch einen direk-
* ten Rücksprung an das Basicprogramm.  BCDE erhalten, andere
* Register zerstört.
GETFLG   INX   H         der Wert (+1, 0, oder -1)
         INX   H         steht im letzten Byte
         INX   H         der 4-Byte langen Integervariablen
         MOV   A,M       hole und
         ORA   A         teste Wert
         STA   FLAG
         RNZ             wenn 0,
         POP   H         lösche Rücksprung zu Maschinenprogramm
         RET
* SCRTAB,TABSCR kopieren Daten von Bildschirm zu Hilfsspeicher
* TAB (dort werden Sie von Unterprogramm CONVERT Diagonal ge-
* spiegelt), bzw. von TAB zu Schirm.  Bei Einsprung enthält HL
* die obere Adresse des angesprochenen Bildschirmblocks.  Je
* nach der Stellung des Carry Bits, entsprechen die oberen (Car-
* ry gesetzt) bzw. unteren (Carry aus) Bildzeilen den niederen
* Adressen in TAB, und je nachdem bestimmt sich an welcher Dia-
* gonale am Schluß effektiv gespiegelt wurde. Ferner komplemen-
* tiert SCRTAB die Daten (also die Farben), wenn bit 7 von A
* gesetzt ist. TABSCR maskiert die Daten (damit der Quellbereich
* nicht überschrieben wird), wenn bit 0 von A rückgesetzt ist.
* A wird zerstoert, A bit 0 wird Carrybit bei Ausgang, andere
* Register bleiben erhalten.
SCRTAB   PUSH  B
         PUSH  D
         PUSH  H
         LXI   B,-LINLEN     Abstand zu nächster Bildschirmzeile
         JC    SCTB20        Wenn kein Carry,
         LXI   B,-7*LINLEN   ändere Anfangszeile und
         DAD   B
         LXI   B,LINLEN      ändere Richtung zur nächsten Zeile
SCTB20   LXI   D,TAB
         RAL                 Bit für komplementiere zu Carry
         MVI   A,8           Bytezähler
```

```
STBLP   PUSH  PSW
        SBB   A             Maske für komplementiere oder nicht
        XRA   M
        STAX  D             bewege Daten zu TAB
        DAD   B             nächste Zeile
        INX   D             nächste Stelle in TAB
        POP   PSW
        DCR   A             zähle 8 Bytes
        JNZ   STBLP
        POP   H
        POP   D
        POP   B
        RET
TABSCR  PUSH  PSW
        PUSH  B
        PUSH  D
        PUSH  H
        LXI   B,-LINLEN     Abstand zu nächster Bildschirmzeile
        JNC   TBSC20        Wenn Carry gesetzt,
        LXI   B,-7*LINLEN   ändere Anfangszeile und
        DAD   B
        LXI   B,LINLEN      ändere Richtung zu nächster Zeile
TBSC20  LXI   D,TAB+7
        RRC
        ORA   A             setze A negativ für maskiere nicht
        MVI   A,:FE         Maske und Bytezähler! (wird geshiftet)
TBSLP   PUSH  PSW
        PUSH  B
        MOV   C,A           Maske
        LDAX  D
        JM    TBSC30        maskiere nicht wenn negativ
        ANA   C             nur die
        MOV   B,A           in der Maske
        MOV   A,M           gesetzten
        ORA   C             Bildschirmbits
        XRA   C             werden
        ORA   B             überschrieben!
TBSC30  MOV   M,A           speichere in Bildschirm
        POP   B
        DAD   B             nächste Zeile
        DCX   D             nächste Stelle in TAB
        POP   PSW
        RAL                 zähle 8 Bytes!
        CMC                 wenn die Maske nicht leer war,
        JNC   TBSLP         ist man noch nicht durch!
        JMP   EXIT          Register wiederherstellen und fertig
* CONVERT vertauscht Bitreihen und Bitspalten in dem 8-Byte
* (also 8x8-Bit) Block TAB (dies bedeutet eine Spiegelung der
* Bits in der Diagonalen).  Bei dem verwendeten Verfahren wird
* jedes Byte der Tabelle der Reihe nach als Ausgangsbyte
* genommen und tauscht mit jedem der folgenden Byte ein Bit aus,
* wonach das Ausgangsbyte schon die richtige gespiegelte Endge-
* stalt hat (und die folgenden Bytes schon die gespiegelten Bits
* enthalten, die aus ihren Vorgängern stammen).  Nachdem jedes
* Byte als Ausgangsbyte gedient hat, sind alle richtig gespie-
* gelt.  Alle Register bleiben erhalten.
CONVERT PUSH  PSW
        PUSH  B
        PUSH  D
        PUSH  H
        LXI   H,TAB     Adresse der Tabelle
        MVI   D,7       acht Byte insgesamt
```

```
CON05     PUSH  H         merke dir jetziges Ausgangsbyte
          MOV   E,D       jetziger Zähler
          MOV   A,M       hole Ausgangsbyte
          RLC             behalte linkes Bit (auf Diagonale!)
CON10     RLC             nächstes Bit zu Carry
          RRC             aber verschiebe Daten nicht!
          MOV   B,A       rette Ausgangsbyte
          INX   H
          MOV   A,M       schiebe Carrybit in nächstes
          RAL             Byte im Speicher und
          MOV   M,A       Bit aus diesem Byte
          MOV   A,B       in das Ausgangsbyte
          RAL
          DCR   E         diese Runde fertig?
          JNZ   CON10     nein
          POP   H         ja, speichere geändertes Ausgangsbyte
          MOV   M,A
          INX   H         und nehme neues Ausgangsbyte
          DCR   D         Zähler für nächste Runde, 1 Byte kürzer
          JNZ   CON05
          MOV   A,M       letzte Runde, letztes Ausgangsbyte
          RLC             muß nur rotiert werden
          MOV   M,A
          JMP   EXIT      Register wiederherstellen, fertig.
* INVERT kehrt die Reihenfolge der Bits in A um, andere Register
* bleiben erhalten.
INV00     LDAX  D         Zusatzeinsprung, holt Daten aus Speicher
          DCX   D         (Speicherzeiger DE wird aktualisiert!)
          XRA   C         hier werden eventuell die Farben umge-
          REM             kehrt:  in C steht 0 oder :FF
INVERT    PUSH  H
          MOV   H,A       Eingabe zu H
          MVI   A,:80     das gesetzte Bit dient als Shiftzähler!
          DAD   H         linkes Bit von H zu Carry!
          RAR             und von dort nach rechts zu A
          JNC   $-2       zählt Shifts, 8 Durchläufe der Schleife!
          POP   H
          RET
TAB       RES   8         Hilfsspeicher für Diagonalspiegelung
FLAG      RES   1         Marke für Farbumkehrung oder nicht
```

```
1     REM    HIER EIN BASICPROGRAMM
2     REM    ALS ANWENDUNGSHILFE
3     REM    UND BEISPIEL
10    GOTO 1000
50    REM Hier gehört ein Unterprogramm hin, das die zu
51    REM zeichnende Farbe FARBE an der Stelle XX!,YY! berechnet,
52    REM wobei die Koordinatenwerte 0 dem Mittelpunkt des Bild-
53    REM schirms entsprechen (also XX! läuft von -XMAX/2.0 zu
54    REM +XMAX/2.0 und YY! läuft von -YMAX/2.0 zu +YMAX/2.0!!).
55    RETURN
1000  REM Hier beginnt die Abfrage nach Symmetrie des Bildes
1015  XREF=#300:YREF=XREF+3:CREF=YREF+3:ANTIREF=CREF+3:EQREF=
C       ANTIREF+3
1020  MODE 0:PRINT CHR$(12);TAB(20);"SYMMETRISCHE BILDER"
1049  ARROW$=" "+CHR$(136)+CHR$(137)+" "
1050  CFLAG=0.0:ANTIFLAG=0.0:PRINT "Symmetrie in x";ARROW$;"-x?":
C       GOSUB 10000:XFLAG=FLAG
1060  PRINT "Symmetrie in x";ARROW$;"y?":GOSUB 10000:EQFLAG=FLAG
```

```
1065  IF EQFLAG<>0.0 THEN YFLAG=XFLAG:IF XFLAG=0.0 THEN 1080:
C       GOTO 1100
1070  PRINT "Symmetrie in y";ARROW$;"-y?":GOSUB 10000:YFLAG=FLAG
1075  IF XFLAG<>0.0 OR YFLAG<>0.0 THEN 1100
1080  PRINT "Symmetrie in (x,y)";ARROW$;"(-x,-y)?":GOSUB 10000:
C       CFLAG=FLAG
1085  IF EQFLAG<>0.0 OR CFLAG<>0.0 THEN 1100
1090  PRINT "Symmetrie in x";ARROW$;"-y?":GOSUB 10000:ANTIFLAG=
C       FLAG
1100  PRINT :MODE 6:XMID!=XMAX/2.0:YMID!=YMAX/2.0
1110  XLOW!=0.5:IF XFLAG=0.0 THEN IF (CFLAG=0.0 OR EQFLAG=0.0)
C       THEN XLOW!=-XMID!:IF CFLAG<>0.0 THEN XLOW!=-YMID!+1.0
1150  FOR XX!=XLOW! TO XMID!
1200  IF YFLAG<>0.0 THEN YLOW!=0.5:GOTO 1250
1205  YLOW!=-YMID!
1210  IF ANTIFLAG<>0.0 THEN IF XX!>=(-YMID!) THEN IF XX!<YMID!
C       THEN YLOW!=-XX!
1220  IF CFLAG<>0.0 THEN IF XX!<YMID! THEN YLOW!=-XX!:IF YLOW!>
C       0.0 THEN YLOW!=YLOW!+1.0
1250  YHIGH!=YMID!:IF EQFLAG<>0.0 THEN IF XX!>=(-YMID!) THEN IF
C       XX!<YMID! THEN YHIGH!=XX!
1300  FOR YY!=YLOW! TO YHIGH!
1310  GOSUB 50: REM ermittle FARBE des Punktes
1325  Y=YMID!+YY!+0.5:X=XMID!+XX!+0.5
1400  DOT X,Y FARBE
1450  NEXT:NEXT
1460  CALLM EQREF,EQFLAG:CALLM XREF,XFLAG:CALLM YREF,YFLAG
1470  CALLM CREF,CFLAG:CALLM ANTIREF,ANTIFLAG
1475  REM nur die Programme mit ..FLAG ≠ 0 werden ausgeführt!
1500  END
10000 INPUT "(S[ymmetrisch],A[ntisymmetrisch],W[eder noch])";A$:
C       PRINT
10010 IF A$="S" THEN FLAG=1:RETURN
10020 IF A$="A" THEN FLAG=-1:RETURN
10030 IF A$="W" THEN FLAG=0:RETURN
10040 GOTO 10000
```

Bochum, den 12.1.85 — Gordon Wassermann

FRAKTALE

Fraktale sind geometrische Gebilde, deren Dimension nicht eine ganze Zahl ist, z.B. eine Kurve von Dimension 1,3 oder eine Fläche von Dimension 2,765. So merkwürdig das klingen mag, und so sehr es unserer intuitiven Vorstellung von Dimension widerspricht, solche Gebilde gibt es wirklich, und man kann sie sogar sehr leicht mit einem kurzen Basic-Programm auf dem DAI zeichnen! Wie, das verrate ich unten.

Aber zuerst möchte ich erklären, wie eine solche Dimension überhaupt zu verstehen ist. Dazu müssen wir eine geschickte, also nicht zu naive, Methode verwenden, um die Dimension eines Gebildes zu messen. Die Methode, die ich beschreiben möchte, stammt von dem Mathematiker Felix Hausdorff, und der entsprechende Dimensionsbegriff heißt die Hausdorff-Dimension oder Hausdorff-Besicovitch Dimension.

Stellen wir uns einmal eine zweidimensionale Menge, sagen wir ein quadratisches Gebiet von Seitenlänge 1, und eine eindimensionale Menge, sagen wir ein Geradenstück von Länge 1, in der Ebene vor. Wir wollen versuchen, den **Inhalt** dieser Mengen zu messen, und verfahren dabei wie folgt: wir wählen eine "Meßgenauigkeit" ε, und überdecken unsere Menge mit kleinen Quadraten von Seitenlänge ε, auf möglichst effiziente Weise, d.h. so, daß die Anzahl $N(\varepsilon)$ der benötigten kleinen Quadrate möglichst klein ist. Um das Geradenstück zu überdecken werden wir etwa $1/\varepsilon$ kleine Quadrate brauchen, aber für das zweidimensionale quadratische Gebiet brauchen wir $1/\varepsilon^2$ kleine Quadrate.

Der Gesamtinhalt $I(\varepsilon)$ der kleinen Quadrate, also $N(\varepsilon)$ mal den Inhalt *eines* kleinen Quadrates, ist unsere Abschätzung für den Inhalt des auszumessenden Gebietes. Nehmen wir als den Inhalt eines kleinen Quadrates den üblichen Flächeninhalt ε^2, so erhalten wir für das quadratische Gebiet $I(\varepsilon) = (1/\varepsilon^2)\cdot\varepsilon^2 = 1$, unabhängig davon, was ε war. Dies ist auch der richtige Flächeninhalt von dem großen Quadrat. Wenn wir aber das Geradenstück auf die gleiche Weise ausmessen, so erhalten wir $I(\varepsilon) = (1/\varepsilon)\cdot\varepsilon^2 = \varepsilon$, und wenn wir ε immer kleiner machen so tendiert dies gegen 0. Es überrascht zwar nicht, daß der Flächeninhalt eines Geradenstücks 0 ist, aber sonderlich nützlich ist diese Information auch nicht, zumal nach diesem Verfahren auch ein doppelt so langes Geradenstück die Fläche 0 hat, und somit nicht von dem kürzeren Geradenstück unterschieden wird.

Um für das Geradenstück ein sinnvolleres Maß zu erhalten, lasst uns jetzt den kleinen Quadraten den Inhalt ε zuordnen--dann errechnen wir für unser Geradenstück $I(\varepsilon) = (1/\varepsilon)\cdot\varepsilon = 1$, und das hängt nicht von ε ab und entspricht der tatsächlichen Länge. Für das große Quadrat erhalten wir aber jetzt $I(\varepsilon) = (1/\varepsilon^2)\cdot\varepsilon = 1/\varepsilon$, und dies wird beliebig groß, wenn ε kleiner wird, und tendiert gegen ∞. Jetzt erhalten wir also für das quadratische Gebiet kein gutes Maß.

Beide Male haben wir etwas ähnliches gemacht: wir haben die kleinen Quadrate gezählt, und die Anzahl $N(\varepsilon)$ mit einer Potenz ε^D von der Seitenlänge multipliziert. Mit $D = 2$ erhalten wir für das zweidimensionale Gebiet ein gutes Maß, mit $D = 1$ ist das Ergebnis für die eindimensionale Gerade sinnvoll. Der Exponent D entspricht also *der Dimension* des auszumessenden Gebildes; für ein zweidimensionales Gebiet ist es also sinnvoll die Fläche zu messen, aber für die eindimensionale Gerade sollte man stattdessen die Länge messen, also ein eindimensionales Maß anwenden.

Wir haben außerdem festgestellt, daß man mit einem zu kleinen Exponenten *D* (z.B. Länge von dem Quadrat) den Inhalt ∞ ermittelt, und mit zu großem Exponenten (Fläche von der Geraden) den Inhalt 0.

Wenn man das Meßverfahren ein wenig verfeinert, in dem man zum Überdecken nicht nur Quadrate von Seitenlänge genau ε zuläßt, sondern auch kleinere (von Seitenlänge höchstens ε), und wenn man den Inhalt jedes kleinen Quadrates mit Seitenlänge hoch *D* nimmt, dann gilt die Aussage des letzten Absatzes ganz allgemein: *Für jedes Gebilde in der Ebene gibt es einen "Scheitelwert" D, so daß wenn man den Inhalt mit einem kleineren Exponenten als D mißt, dann erhält man den Wert ∞, und wenn man mit einem größeren Exponenten als D mißt, dann erhält man den Wert 0.* Nur mit Exponenten genau *D* hat man überhaupt Hoffnung, ein endliches Ergebnis ungleich Null zu erhalten. Diese Zahl *D* kann man als die **Dimension** des Gebildes definieren. Für unser Quadrat und unser Geradenstück ist diese Dimension, wie zu erwarten, 2 bzw. 1.

Das Erstaunliche ist aber, daß die so definierte Dimension nicht immer eine ganze Zahl ist! Betrachten wir z.B. einmal ein Geradenstück von Länge *L*. Wir wollen dieses Geradenstück jetzt ersetzen durch einen Zug aus vier Geradenstükken nebenstehender Gestalt, wobei jedes der vier neuen Geradenstücke die Länge *L*/3 hat. Jetzt ersetzen wir jedes der neuen, kürzeren Geradenstücke wieder durch eine ähnliche, nur verkleinerte, Folge von vier Geradenstükken von ein Drittel der Länge, und wiederholen dieses Verfahren immer wieder. Die dabei entstehenden Kurven enthalten immer mehr, aber immer kürzere Segmente (deren Gesamtlänge bei jeder Stufe um den Faktor 4/3 wächst!), und im Grenzwert konvergieren diese Kurven gegen eine sehr stark wackelnde aber stetige Kurve *K* von unendlicher Länge. Das Ergebnis sieht schöner aus, wenn man nicht mit **einem** Geradenstück beginnt, sondern mit dreien, die ein gleichschenkeliges Dreieck bilden. Dann besteht die Grenzkurve aus drei Kopien von *K*, und erinnert stark an eine Schneeflocke, weshalb sie auch die **Koch'sche Schneeflocke** (nach ihrem Erfinder Helge von Koch) genannt wird.

Bild 1: Generator der Koch'schen Kurve

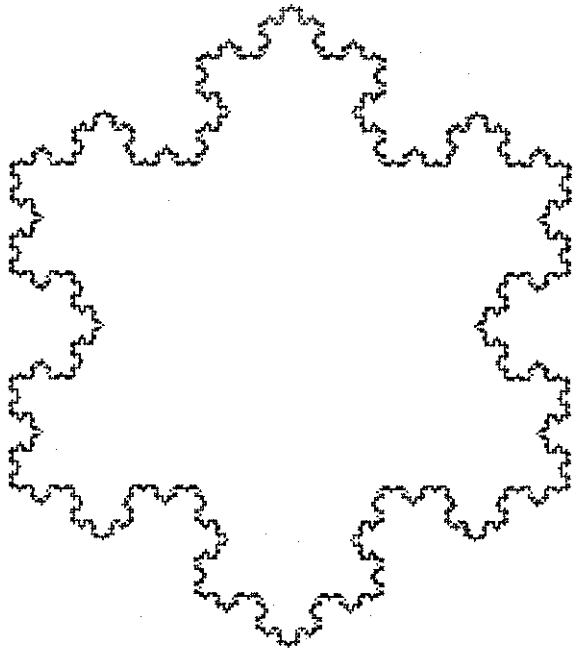


Bild 2: die Koch'sche Schneeflocke

Die Dimension von der Koch'schen Schneeflocke, oder was das gleiche ist, die Dimension von *K*, ist leicht zu bestimmen. Beim ersten Schritt in der Erzeugung von *K* ersetzen wir die Anfangsgerade durch vier Geradenstücke von 1/3 der Größe; danach machen wir mit diesen kürzeren Geraden genau das gleiche, das

wir mit der ursprünglichen Geraden gemacht haben, und wiederholen das unendlich oft, so daß jedes der vier Geradenstücke schließlich zu einer Kopie der ganzen Kurve K wird, nur um den Faktor 3 verkleinert. Also K besteht aus vier Teilstücken, die Kopien von K sind, nur 1/3 so groß (man sagt, K ist *selbstähnlich*). Nun, angenommen, wir messen K aus mit Quadraten von Seitenlänge ε, und brauchen davon $N(\varepsilon)$ Stück; jedes davon zählen wir mit Inhalt ε^D. Um jedes der vier Teilstücke von K zu überdecken, bräuchten wir dann die gleiche Anzahl $N(\varepsilon)$ Quadrate von Seitenlänge $\varepsilon/3$, und um K zu überdecken bräuchten wir $4N(\varepsilon)$ davon. Wir haben also $I(\varepsilon/3) = 4N(\varepsilon)(\varepsilon/3)^D = 4I(\varepsilon)/(3^D)$. Damit also $I(\varepsilon)$ für $\varepsilon \longrightarrow 0$ nicht gegen 0 oder ∞ geht, brauchen wir $4/(3^D) = 1$, also $4 = 3^D$, oder $\log 4 = D \log 3$, d.h. $D = \log 4/\log 3 = 1.26186$; dies ist die Dimension von K!

Listing 1 unten ist ein Basic Programm, das die Koch'sche Schneeflocke zeichnet. Es ist sehr kurz, weil die Koch'sche Schneeflocke, obwohl sehr kompliziert aussehend, nach einem sehr einfachen Schema erzeugt wird, nämlich durch das wiederholte Ersetzen von Geraden durch eine feste andere, aus mehreren Geraden bestehende Figur. Diese Figur heißt der *Generator* der Schneeflocke; das Dreieck, mit dem man die Konstruktion beginnt, heißt der *Initiator*.

Programmiertechnisch interessant an diesem Programm ist die Tatsache, daß das Unterprogramm bei Zeile 50, welches Geradenstücke durch den Generator ersetzt, um die Wiederholung dieses Einsetzens zu erreichen sich selber rekursiv aufruft, und dies obwohl rekursive Unterprogramme in DAI-Basic normalerweise nicht möglich sind. Wieso ist das denn hier möglich? Nun, die Tatsache alleine, daß ein Programm sich selber aufruft, bereitet Basic keine Schwierigkeiten, da die Rücksprungadressen auf dem Stapel abgelegt werden und so automatisch auseinandergehalten werden. Das Problem liegt vielmehr darin, daß alle Variablen in Basic global sind, so daß die verschiedenen Instanzen eines rekursiven Programms sich gegenseitig die Werte der von ihnen verwendeten Variablen zerstören. Diese Schwierigkeit wird hier umgangen durch Verwendung eines Arrays, um einen Datenstapel zu simulieren; die Daten, die vom Unterprogramm geändert werden (das sind die Endpunkte der in der Kurve enthaltenen Geradenstücke), werden nicht in Variablen gehalten, sondern auf diesem Stapel. Die Variable K dient dabei als Stapelzeiger.

Zum Programmablauf im Einzelnen: Die Arrays X und Y bilden den simulierten Stapel, und enthalten die Liste der Endpunkte der Geradenstücke, aus denen die Annäherungen zur Koch'schen Kurve bestehen; in Zeilen 300-340 werden diese Arrays auf den Initiator (also ein gleichschenkliges Dreieck) initialisiert. In Zeile 360 wird Zeile 50 dreimal aufgerufen, um das wiederholte Einsetzen des Generators für die drei Seiten des Dreiecks auszuführen. Das Programm kann natürlich das Einsetzen des Generators nur endlich oft wiederholen. Dieses Einsetzen wird, wie gesagt, vom Unterprogramm in den Zeilen 50-110 bewerkstelligt, und die Variable LEVEL mißt die Rekursionstiefe, zu der dieses Unterprogramm zuletzt aufgerufen wurde. LEVEL wird vor Beginn auf die gewünschte Anzahl der Wiederholungen des Einsetzens eingestellt, und wird bei jedem rekursiven Aufruf von Zeile 50 dekrementiert. Bei LEVEL = 0 ruft das Unterprogramm sich nicht mehr weiter auf.

Das Unterprogramm bei Zeile 50 verarbeitet jeweils das letzte Geradenstück, daß in den Arrays X und Y angegeben ist, also das Geradenstück, dessen Endpunkte durch die letzen beiden

X und Y Werte gegeben werden. Falls LEVEL noch nicht 0 ist (ab Zeile 70) wird die letzte Eintragung von X und Y ersetzt durch die vier neuen Endpunkte der Generatorkopie, die hier eingesetzt wird, und das Programm ruft sich dann vier Mal selber auf (Zeile 110), um die vier neuen Strecken weiterzuverarbeiten. Falls aber beim Aufruf von Zeile 50 schon genügend oft wiederholt wurde, d.h. LEVEL = 0, dann wird (in Zeile 60) die letzte Strecke einfach auf dem Bildschirm gezeichnet, und dann durch heruntersetzen des Stapelzeigers K aus X und Y gelöscht.

Für die Koch'sche Schneeflocke ist LEVEL = 5 ein guter Eingangswert, da die Geraden der fünften Stufe schon kürzer sind als die Breite eines Bildschirmpunktes. Da es aber auch lehrreich ist, die Entstehung der Kurve zu verfolgen, wird sie nicht nur in ihrem Endstadium gezeigt, sondern die Schleife in Zeile 225 läßt alle Stufen nacheinander zeichnen; nach dem Zeichnen wartet das Programm (Zeile 365) auf Betätigung der Leertaste, bevor es die nächste Stufe zeigt.

Das Verfahren zur Erzeugung der Schneeflocke läßt sich sehr leicht abwandeln und verallgemeinern, in dem man einen anderen Generator und vielleicht auch einen anderen Initiator verwendet, aber ansonsten genauso verfährt, wie oben. Auch die Ausrichtung des Generators bezüglich der ersetzten Gerade läßt sich variieren, und braucht auch nicht wie oben bei allen Geradenstücken ein und derselben Kurve immer gleich zu bleiben. Auch das Basicprogramm läßt sich leicht für solche Änderungen anpassen, durch Änderung der Zeilen 70-110 und 300-360. Ein Beispiel dafür wird durch Listing 2 gegeben. Hier ist der Initiator bloß eine Gerade _, und auch der Generator ist denkbar einfach: ⩘ . Er besteht aus zwei Geraden im rechten Winkel zueinander, jede $1/\sqrt{2}$ so lang wie die zu ersetzende Gerade. Läßt man die Spitze des Generators entlang der Kurve immer abwechselnd nach rechts und nach links zeigen, und läßt man bei jeder weiteren Annäherungsstufe den Generator am Anfang der Kurve auch abwechselnd nach rechts und links zeigen, so erhält man eine Peanokurve, d.h. eine raumfüllende Kurve, die ein Dreieck ganz ausfüllt. Diese Kurve hat Hausdorffdimension 2. Das Programm in Listing 2 zeichnet sie.

Allerdings, in jeder Annäherungsstufe zu dieser Peanokurve berühren sich die Teilgeraden an den Eckpunkten eines quadratischen Gitters, so daß man nur das Gitter sieht und nicht mehr den genauen Verlauf der Kurve. Um diesen Verlauf wieder sichtbar zu machen, schneidet das Programm die Ecken, um die die Kurve läuft, stumpf ab, indem es nur die mittlere Hälfte jeder Teilstrecke zeichnet und die aufeinanderfolgenden gekürzten Teilstrecken durch schräg verlaufende Linien verbindet. Deshalb ist die Zeile 60 aus Listing 1 durch das kompliziertere Zeichenverfahren in Zeilen 60-63 aus Listing 2 ersetzt worden.

Man beachte auch die Variablen SIGN und LSIGN, die in Listing 2 dafür sorgen, daß die Ausrichtung des Generators wechselt, sowohl bei aufeinanderfolgenden Strecken einer Stufe (SIGN), wie auch bei der ersten Strecke jeder neuen Stufe (LSIGN).

Macht man bei diesem zweiten Beispiel, der Peanokurve, nur eine kleine Änderung, und zwar, daß der Generator am Anfang jeder Annäherungsstufe jetzt *immer* nach rechts zeigen soll (aber bei aufeinanderfolgenden Strecken dieser Stufe wie zuvor immer die Ausrichtung wechselt), so erhält man zwar wieder eine

raumfüllende Kurve, aber sie füllt nicht mehr ein Dreieck aus, sondern die nebenstehende wesentlich kompliziertere Figur, den sogenannten *Drachen*. Um das Programm in Listing 2 den Drachen zeichnen zu lassen, muß man nur die Anweisungen LSIGN=-LSIGN aus Zeile 110 entfernen. Allerdings empfiehlt es sich auch, die Variablen XOLD und YOLD, insbesondere Zeile 363, aus dem Programm zu entfernen, und die Zeilen 60-63 durch Zeile 60 aus Listing 1 zu ersetzen; dadurch werden die Ecken zwar nicht mehr abgeschnitten und die Kurven haben Selbstberührungspunkte, aber das ist in diesem Fall wegen der interessanteren Struktur der Kurve nicht so störend, und die Zeichengeschwindigkeit wird wesentlich erhöht.

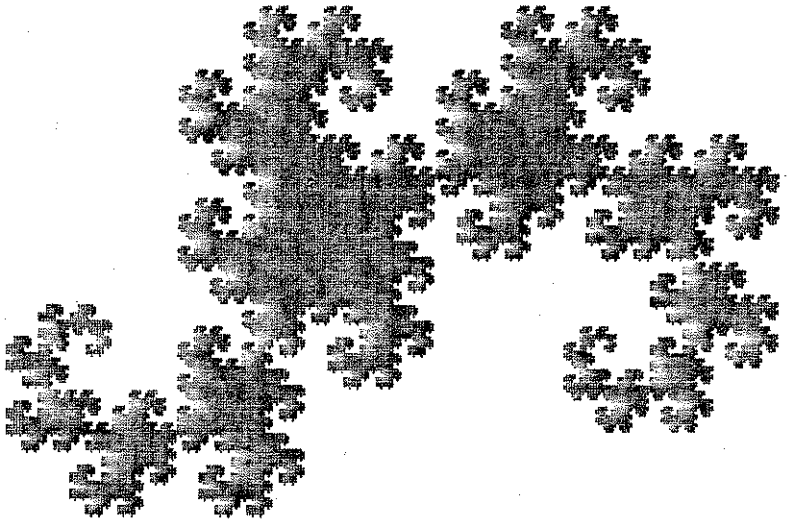


Bild 3: der Drachen

Am Beispiel des Drachens sieht man, wie einfach es ist, mit diesen fraktalen Kurven zu experimentieren. Ich wünsche Ihnen noch viel Spaß beim Zeichnen neuer schöner Fraktale.

Listing 1: **Koch'sche Schneeflocke**
Vor dem eintippen: IMPFPT: IMPINT I-L

```
5     GOTO 200
50    REM Unterprogramm um Geraden durch
55    REM den Generator zu ersetzen
60    IF LEVEL=0.0 THEN DRAW X(K),Y(K) X(K-1.0),Y(K-1.0) 21:K=K-
        1:RETURN
70    X=X(K):Y=Y(K):XX=X(K-1.0):YY=Y(K-1.0)
80    DX=XX-X:DY=YY-Y
90    X(K+1.0)=(X+XX)/2.0-FACTOR*DY:Y(K+1.0)=(Y+YY)/2.0+FACTOR*
        DX
100   DX=DX/3.0:DY=DY/3.0:X(K)=XX-DX:Y(K)=YY-DY:X(K+2.0)=X+DX:
        Y(K+2.0)=Y+DY:K=K+3:X(K)=X:Y(K)=Y
110   LEVEL=LEVEL-1:GOSUB 50:GOSUB 50:GOSUB 50:GOSUB 50:LEVEL=
        LEVEL+1:RETURN
200   CLEAR 20000
210   FACTOR=1.0/(2.0*SQR(3.0))
220   DIM X(255.0),Y(255.0)
225   FOR LEVEL=0 TO 5
230   MODE 6:COLORG 0 3 5 10
```

```
300   REM Hier wird der Initiator erstellt
310   X=60.0*SQR(3.0):X(0.0)=XMAX/2.0+X:Y(0.0)=YMAX/2.0-60.0
320   X(1.0)=XMAX/2.0:Y(1.0)=YMAX/2.0+120.0
330   X(2.0)=XMAX/2.0-X:Y(2.0)=Y(0.0)
340   X(3.0)=X(0.0):Y(3.0)=Y(0.0):K=3
360   GOSUB 50:GOSUB 50:GOSUB 50
365   CALLM #D6DA: REM PAUSE
370   NEXT LEVEL
```

Listing 2: Peano Kurve
Vor dem eintippen: IMPFPT: IMPINT I-L

```
5     GOTO 200
50    REM Unterprogramm um Geraden durch
55    REM den Generator zu ersetzen
60    IF LEVEL>0.0 THEN 70:X=X(K):Y=Y(K):XX=X(K-1.0):YY=Y(K-1.0):
        DX=(XX-X)/4.0:DY=(YY-Y)/4.0
63    X=X+DX:Y=Y+DY:XX=XX-DX:YY=YY-DY:DRAW XOLD,YOLD X,Y 21:DRAW
        X,Y XX,YY 21:XOLD=XX:YOLD=YY:K=K-1:RETURN
70    X=X(K):Y=Y(K):XX=X(K-1.0):YY=Y(K-1.0)
80    DX=(XX-X)/2.0:DY=(YY-Y)/2.0
105   X(K)=X+DX-SIGN*DY:Y(K)=Y+DY+SIGN*DX:K=K+1:X(K)=X:Y(K)=Y
110   LEVEL=LEVEL-1:LSIGN=-LSIGN:SIGN=LSIGN:GOSUB 50:SIGN=-LSIGN:
        GOSUB 50:LEVEL=LEVEL+1:LSIGN=-LSIGN:RETURN
200   CLEAR 20000
205   FACTOR=1.0/(2.0*SQR(3.0))
220   DIM X(255.0),Y(255.0)
225   FOR LEVEL=0 TO 17
230   MODE 6:COLORG 0 3 5 10
300   REM Hier wird der Initiator erstellt
310   X=128.0:X(0.0)=XMAX/2.0+X:Y(0.0)=0.0
320   XOLD=XMAX/2.0-X:YOLD=0.0:X(1.0)=XOLD:Y(1.0)=YOLD
330   K=1
337   SIGN=1.0
340   LSIGN=1
360   GOSUB 50
363   DRAW XOLD,YOLD X(0.0),Y(0.0) 21
365   CALLM #D6DA: REM PAUSE
370   NEXT LEVEL
```

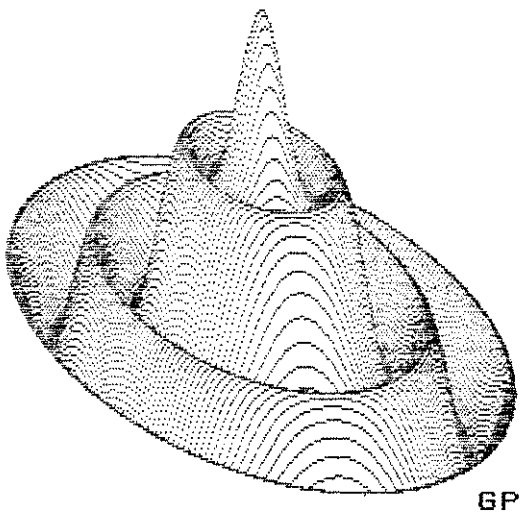
Literatur: B. Mandelbrot, *The Fractal Geometry of Nature*, W.H. Freeman and Company, New York 1983.

Bochum, den 28. 4. 1985 G. Wassermann

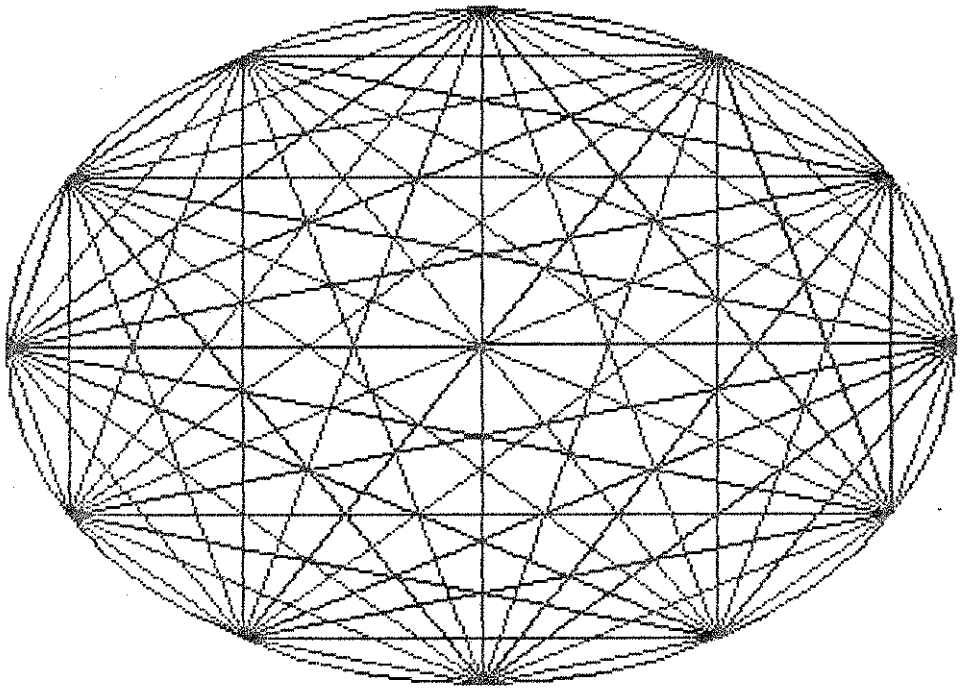
3D-Plot von Andreas Koldehoff

Dieses kleine Programm erzeugt die bereits in der vorletzten Ausgabe ausgedruckten Graphiken (siehe GP 16,2 und GP 17,2). Es handelt sich hierbei um ein in der Zeitschrift CREATIVE COMPUTING 1/83 veröffentlichtes, für den APPLE geschriebenes, Programm.
Die Idee des Programms ist es, eine Funktion um die y-Achse rotierend darzustellen. Deshalb ist es nur sinnvoll gerade Funktionen, also f(x)=f(-x), in Zeile 30 zu verwenden.
Ein Wort zur Rechenzeit: Da nicht nur Punkte gesetzt, sondern auch evtl. später nicht sichtbare Punkte gelöscht werden, ergibt sich eine recht lange Programmlaufdauer. Für eine komplette Funktionendarstellung werden ca. 90 min benötigt (ohne AMD !). Um z.B. einen ersten Eindruck über das Aussehen einer Funktion in der 3D-Darstellung zu gewinnen, kann man in Zeile 1170 ein 'STEP 5' anfügen. Die Programmlaufzeit verkürzt sich dem entsprechend.
Hier einige Funktionen, die befriedigende Ergebnisse zeigen :
1) f(x)=cos(4x)+20/(x*x+3)
2) f(x)=cos(x)+cos(2x)+cos(5x)
3) f(x)=cos(2x)+cos((x+BB)/16) <== siehe GP 16,2 unten

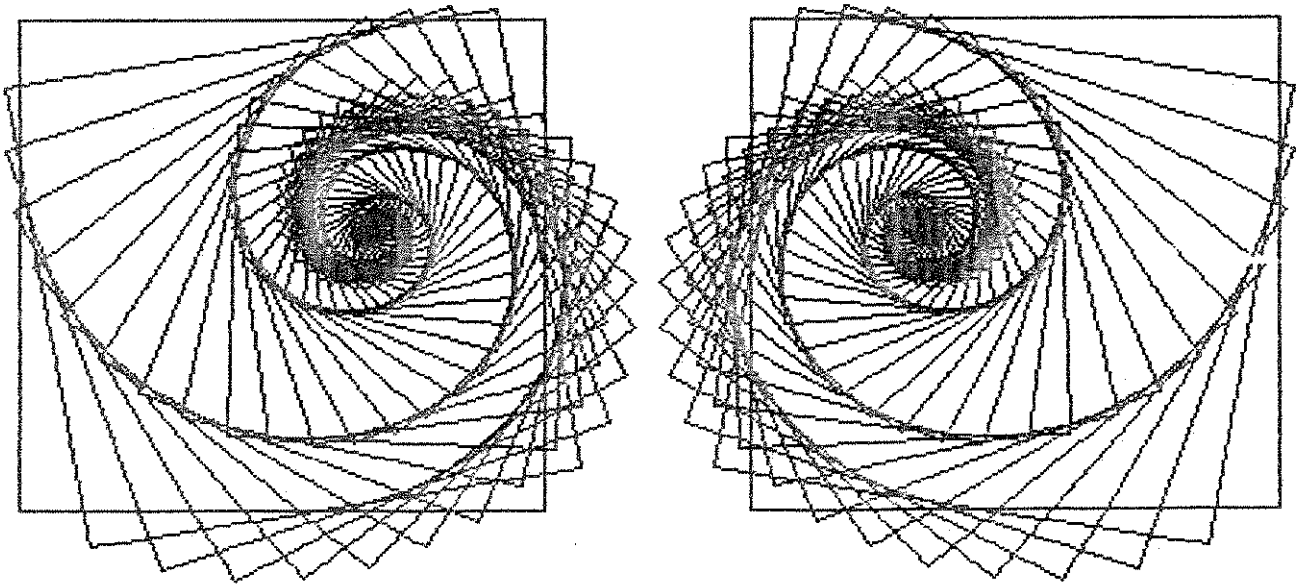
```
*IMPINT
*LIST
10     GOTO 1000
20     REM DEF FN R(Q)
30     R!=COS(4.0*Q!)+20.0/(Q!*Q!+3.0):RETURN
40     X=BB+(H!/B!)+E!
50     Y=DD!-(H!/B!)+F!
60     IF Y<0 THEN RETURN
70     IF Y>255 THEN DRAW X,255 X,0 20:RETURN
80     DOT X,Y 21
90     DRAW X,Y-1 X,0 20:REM Hintergrund loeschen
100    RETURN
1000   MODE 0:COLORT 8 0 0 0
1010   PRINT :PRINT :PRINT
1020   PRINT CHR$(12);"3D-PLOT":PRINT "=======":PRINT
1030   PRINT "Funktion: ";:LIST 30:PRINT
1040   PRINT "Um die Graphik vertikal zu vergroessern oder zu
       verkleinern,"
1050   PRINT "geben Sie bitte eine Zahl zwischen -40 und +40
       ein !"
1060   INPUT "(20 ist typisch) Zahl ";N1!
1070   PRINT :PRINT
1080   PRINT "Um die Graphik auf dem Bildschirm vertikal zu
       zentrieren,"
1090   PRINT "geben Sie bitte eine Zahl zwischen -50 und +150
       ein !"
1100   INPUT "(90 ist typisch) Zahl ";N2!
1110   A=144:B!=2.25:C!=N1!
1120   D!=3.27E-2:E!=160.0:F!=N2!*0.9
1130   REM
1140   MODE 6:COLORG 8 3 5 10
1150   FOR H!=-A TO A STEP B!
1160   AA=0.5+SQR(A*A-H!*H!)
1170   FOR BB=-AA TO AA
1180   Q!=SQR(BB*BB+H!*H!)*D!
1190   GOSUB 30:DD!=R!*C!
1200   GOSUB 40
1210   NEXT:NEXT
1220   CALLM #D6DA:END
*RUN
```



```
100        OSTEREIER     H. Klaskala, 27.3.85
110
120    IMP INT
130
140    MODE 2:GOSUB 150:MODE 4:GOSUB 150:MODE 6:GOSUB 150:GOTO 140
150    XM=XMAX/2:YM=YMAX/2:PP!=PI+PI:XMM=XM:YMM=YM:COLORG 1 5 10 15
160    IF RND(0.0)>0.7 THEN XMM=XM-RND(XM/2):GOTO 180
170    IF RND(0.0)>0.55 THEN YMM=YM-RND(YM/2)
180    I=0:FOR W!=0 TO PP! STEP PP!/48:I=I+1
190    X=XM+XMM*COS(W!):Y=YM+YMM*SIN(W!)
200    IF W!=0.0 THEN XX=X:YY=Y
210    DRAW X,Y XX,YY 21+I MOD 3:XX=X:YY=Y:NEXT
220    N=5+RND(YMAX/10):PN!=PP!/N
230    FOR W!=0 TO PP! STEP PN!
240    X=XM+XMM*COS(W!):Y=YM+YMM*SIN(W!)
250    FOR WW!=W! TO PP!-PN!/2.0 STEP PN!
260    XX=XM+XMM*COS(WW!):YY=YM+YMM*SIN(WW!)
270    DRAW X,Y XX,YY 21+RND(3):NEXT:NEXT
280    FOR I=1 TO 60:COLORG 1 RND(16) RND(16) RND(16):NEXT
290    COLORG 1 15 15 15:WAIT TIME 50+XMAX:RETURN
```



Hardcopy ist graustufenlos



DREHENDES QUADRAT

H.Klaskala, 27.3.85

```
100     DREHENDES QUADRAT   H.Klaskala, 27.3.85
110
120   IMP INT
130
140   CLEAR 1000
150   MODE 6
160   COLORG 0 5 10 15
170   N=4
180   DIM X!(N),Y!(N)
190   FOR I=0 TO N:READ X!(I),Y!(I):NEXT
200   GOSUB 3000
210   FOR J=0 TO 80
220   PX!=161.0:PY!=141.0:Z!=0.95:GOSUB 2000
230   PX!=161.0:PY!=141.0:Z!=0.17:GOSUB 1000
240   GOSUB 3000
250   NEXT J
260   IF GETC=0 THEN 260
270   GOTO 100
1000  REM Drehung (PX,P,Z) Z=Winkel in rad
1010  FOR I=0 TO N
1020  UX!=X!(I)-PX!:UY!=Y!(I)-PY!
1030  UR!=SQR(UX!*UX!+UY!*UY!):UW!=ATN(UY!/UX!)
1040  IF UX!<0.0 THEN UW!=UW!+PI
1050  UW!=UW!+Z!
1060  X!(I)=PX!+UR!*COS(UW!):Y!(I)=PY!+UR!*SIN(UW!)
1070  NEXT I
1080  RETURN
2000  REM Zentrische Streckung (PX,PY,Z) Z=Streckfaktor
2010  FOR I=0 TO N:X!(I)=PX!+(X!(I)-PX!)*Z!:Y!(I)=PY!+(Y!(I)-PY!)
      *Z!:NEXT I
2020  RETURN
3000  REM ZEI
3010  FOR I=0 TO N-1:DRAW X!(I),Y!(I) X!(I+1),Y!(I+1) 23:NEXT
3020  RETURN
4000  DATA 35,35,220,35,220,220,35,220,35,35
```

GRAFIK-CHAOS

H.Klaskala, 27.3.85

```
100     GRAFIK-CHAOS    H.Klaskala, 27.3.85
110
120   Ein kurzes und einfaches, aber umwerfendes Programm :
130   Der Video-RAM wird mit "zufaelligen" Werten voll-
140   geschrieben. Ergebnis: einfach unbeschreiblich
150
160   X=#1000
170   READ Y:IF Y<>999 THEN POKE X,Y:X=X+1:GOTO 170
180   CALLM #1000
190   DATA 6,#B3,#E,#50,0,0,#78,#E6,#BF,#F6,#B0,#47,#14,#81,#AA,#
      1C,#83,#57,2,3,3,3,3,3,#C3,6,#10,999
```

P O T - P O U R R I

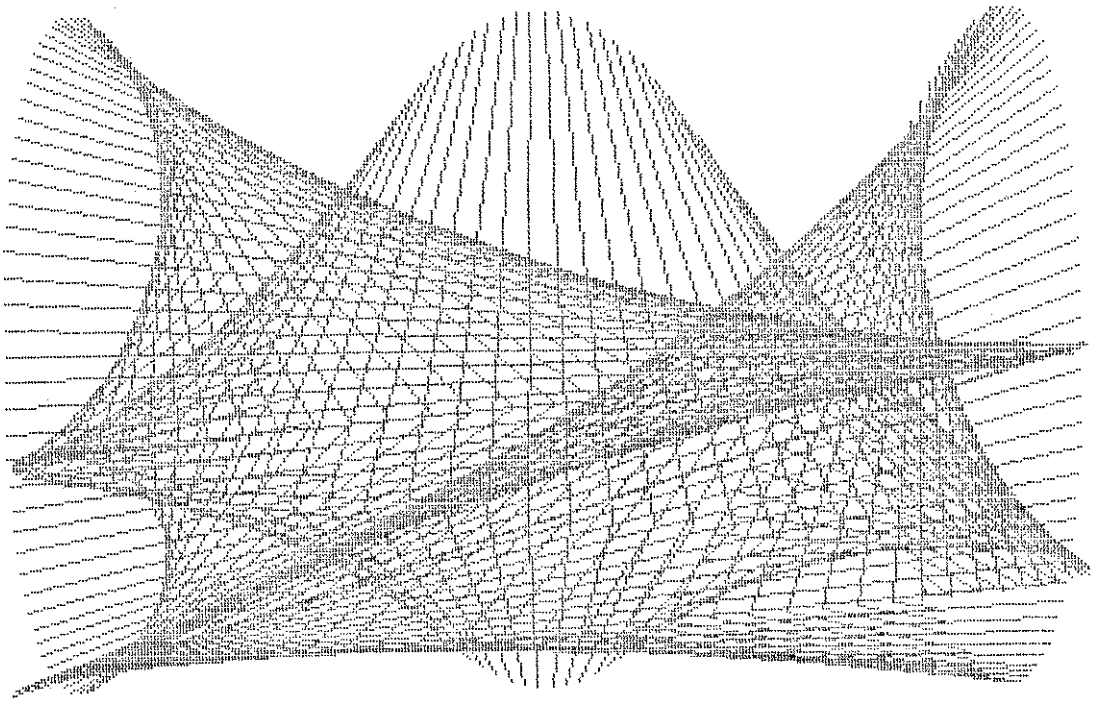
```
100   CLEAR 5000:ENVELOPE 0 15,100;
110   PRINT CHR$(12);"POT-POURRI  -  MORCEAU SUIVANT":WAIT TIME 20
120   WAIT TIME 50:READ N,NN:IF N=0 THEN END
130   N=360/N
140   DIM A(250),B(250)
150   FOR I=1 TO NN:READ A(I),B(I):NEXT I
160   FOR I=1 TO NN
170   SOUND 0 0 15 0 FREQ(A(I))
180   IF B(I)>1 THEN B(I)=B(I)-1:WAIT TIME N:GOTO 180
190   IF B(I)=0 THEN WAIT TIME 1:GOTO 210
```

Lissayous Figuren
Bearbeitet: Uwe Wienkop

Das untenabgedruckte Programm zur Erzeugung von Lissayus Figuren wurde unter Verwendung einiger XBASIC Befehle und des Unterstreichungsstrichs '_' in Variablen geschrieben. Den Strich kann man einfach weglassen. Die XBASIC Befehle sind wohl ohne Schwierigkeiten in normale BASIC Kommandos zu überführen: Der Anweisungsteil zwischen REPEAT UNTIL wird solange ausgeführt, bis die Bedingung nach UNTIL erfüllt ist.

Das Programm kann in beliebigen Graphik-Modi (auch MODE 8) laufen. Hierzu sind nur die MODE Anweisungen in den Zeilen 130 und 140 anzupassen.

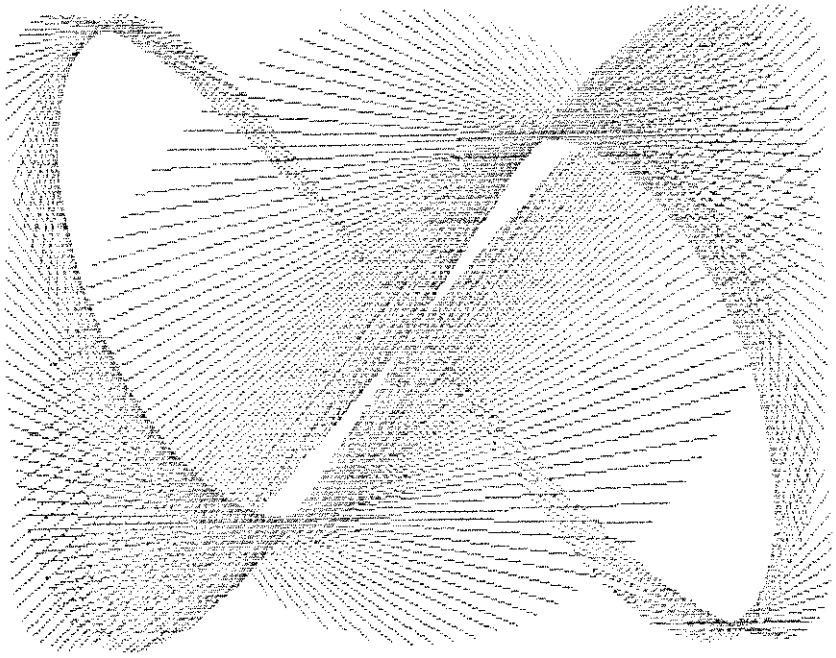
☞ Das Programm stammt ursprünglich vom ATARI 520 ST in der Programmiersprache C. Der Unterschied zu unserem DAI ist bei diesem Programm bemerkenswert:
Die Initialisierung des Arrays dauerte nur ca. 4-5 sec (DAI mit AMD: ~27 sec) und die Erzeugung eines Bildes im Schnitt 1.5 sec und das alles, obwohl das untenstehende Programm nur ca. 47% der Anzahl der Koeffizienten berechnet und nur ca. 70% der Anzahl der Linien des Ataris gezeichnet werden. Allerdings muß man berücksichtigen, daß es sich um einen Vergleich Compiler⇔Interpreter handelt, der nicht ganz fair ist.



```
100   REM --------- Lissayous Figuren ------------
110   CLEAR 12700
120   ANZ_FUNKT=13:ANZ_LINIEN=242:DIM FUNC(ANZ_FUNKT-1,ANZ_LINIE
      N)
130   MODE 6:GOSUB 1000:XM=XMAX:YM=YMAX
140   REPEAT:MODE 6
150   I=RND(ANZ_FUNKT):REPEAT:J=RND(ANZ_FUNKT):UNTIL I<>J
160   K=RND(ANZ_FUNKT):REPEAT:L=RND(ANZ_FUNKT):UNTIL K<>L
170   FOR M=0 TO ANZ_LINIEN:DRAW FUNC(I,M),FUNC(J,M)*YM/XM FUNC(
      K,M),FUNC(L,M)*YM/XM 21:NEXT
180   UNTIL GET$<>0:END
190
200
980   REM Berechnung der Koeffizienten:
990
1000  MITTE!=XMAX/2:V!=2*PI/ANZ_LINIEN:XM!=XMAX
1010  FOR X=0 TO ANZ_LINIEN:X!=X
1020  FUNC(0,X)=ABS(2*(X!-ANZ_LINIEN/2)*MITTE!/ANZ_LINIEN)+MITTE
      !
1030  FUNC(1,X)=SIN(V!*X!)*MITTE!+MITTE!
1040  FUNC(2,X)=XM!-FUNC(1,X)
1050  FUNC(3,X)=COS(V!*X!)*MITTE!+MITTE!
1060  FUNC(4,X)=XM!-FUNC(3,X)
1070  FUNC(5,X)=SIN(2*V!*X!)*MITTE!+MITTE!
1080  FUNC(6,X)=XM!-FUNC(5,X)
1090  FUNC(7,X)=COS(2*V!*X!)*MITTE!+MITTE!
1100  FUNC(8,X)=XM!-FUNC(7,X)
1110  FUNC(9,X)=SIN(3*V!*X!)*MITTE!+MITTE!
1120  FUNC(10,X)=XM!-FUNC(9,X)
1130  FUNC(11,X)=COS(3*V!*X!)*MITTE!+MITTE!
1140  FUNC(12,X)=XM!-FUNC(11,X)
1150  NEXT:RETURN
```

```
   29 Lines            15 Symbols
    1 References      118 References
  826 Bytes Text      130 Bytes Symbols

   48 Commands
```



Darstellen und Ausdrucken von Funktiongraphen

Hintergrund
In der Analysis (Oberstufenunterricht an der Schule) müssen oft Funktionsgraphen - bezogen auf die Längeneinheit 1cm (2cm) - gezeichnet werden. Ziel der Programmentwicklung war es, dies mit dem Computer in Verbindung mit dem Drucker FX-80 zu ermöglichen, wobei

1. der (die) Funktionsgraph(en) nebst Achsenkreuz mit Beschriftung sowohl auf dem Bildschirm dargestellt, als auch vom Drucker (mit einer max. Abweichung von 0,5 mm) ausgegeben werden sollte
2. der Ausdruck mit der von den Hardcopy-Programmen gewohnten Schnelligkeit erfolgen sollte
3. Ein x - und y - Wertebereich von ca 15 cm (bei LE: 1cm) möglich sein sollte.

Wegen 2 werden alle Druckkopfnadeln (bis auf eine) gleichzeitig verwendet, also 8 übereinanderliegende Punkte gleichzeitig ausgedruckt. Da der Nadelabstand 3/216 Zoll beträgt, ergibt dies 72 Punkte je Zoll bzw. 28,34 Punkte je cm, womit die 'Positioniergenauigkeit' der 1cm-Marken etwa 0,3mm beträgt. Wegen 1. und 2. werden auf dem Bildschirm ca 425 Punkte in y-Richtung benötigt. Dies wird dadurch gelöst, das die Farbe der Punkte verwendet wird (z.B. Bildschirm y=0 und Farbe 1 enspricht y=0 für den Drucker, y=0 und Farbe 2 y=1 für den Drucker). Um auch für die x-Richtung einen großen Bereich zu ermöglichen, wurde MODE 8 zugrundegelegt (Das Programm läßt sich jedoch mit eingeschränktem x-Wertebereich auch in MODE 6 betreiben).

Zum Programm:
Es besteht aus einem Basicteil, dem Maschinenprogramm SYMBOL, das die Achsenkreuz-Beschriftung durchführt (die Ziffern erscheinen auf dem Bildschirm in y-Richtung 'geschrumpft', sind aber noch lesbar!) und dem Maschinenprogramm COPY für den Drucker. Die Maschienen-Programme beginnen erst ab #4B0, damit sowohl das unter NP26 angegebene MODE8 Maschinenprogramm *(ich selbst verwende die neuen BASIC-Eproms von H.Tegethoff)*, als auch das unter NP12 angegebene Maschinen-Programm 'Funktionseingabe für BASIC' untergebracht werden kann. Das letztere funktioniert bei mir leider nicht richtig (nach der ersten Eingabe grundsätzlich 'SYNTAX ERROR', nach der zweiten Ausstieg aus dem BASIC).
Das M-Programm COPY übergibt die Daten an den Drucker über die Parallel-Schnittstelle. Soll dies anders gestaltet werden, so ist das Unterprogramm PRINT (Zeile 132-137) entsprechend abzuändern. Soll *anstatt des FX-80* ein Drucker verwendet werden, bei dem die oberste Nadel durch das niederwertigste Bit angesteuert wird, so müßte es reichen, statt der Zeilen 69, 85 und 87 69 YLOOP MVI C,128 - 85 RAR - 87 JNZ BLOCLO einzugeben. Ferner sind gegebenenfalls die Steuerzeichen in Zeile 39 - 66 abzuändern. Der Drucker muß in Plottmodus arbeiten!
Zum Basicteil:
Alle Variablen mit ! sind vom Typ real, sonst integer.
Lücken zwischen den Punkten des Funktionsgraphen werden durch senkrechte Strecken au gefüllt.'Genauer' (und schneller!) wird es ohne diese; dann ist 317 RETURN einzufüge
Zur Eingabe und Inbetriebnahme:
Die Zeilen 10-60 beziehen sich auf das KEN-DOS-System. Für die Kassetten-Benutzer gibt es u.a. die folgende Möglichkeit:

1. BASIC-Teil ohne die Zeilen 10-60 eingeben und abspeichern
2. Maschinen-Programme eingeben und ebenfalls abspeichern
3. NEW:CLEAR #850-#2EC:POKE9B,#50:POKE9C,8:CLEAR#100
4. BASIC-Teil und die Maschinen-Programme laden
5. UT
   - Adresse A vom Ende der Variablentabelle mit Hilfe von S... ermitteln (diese ste unter 2A3 und 2A4)
   - W29B A Programmname

Das Programm wird dann mit UT, R geladen und anschließend mit B, RUN in Betrieb geno men.

Anmerkung: Das Programm SYMBOL läßt sich durch weitere Zeichen (8 Punkte breit, Höhe beliebig) außer '.' und dem Leerzeichen ergänzen, in dem man nach der Zeile 195 weit re DATA-Zeilen einfügt; dabei muß für jede DATA-Zeile gelten:
Erst das zugeordnete ACII-Zeichen, dann die Inhalte für jedes Byte, zum Schluß die Zahl 1.

GP 25,1

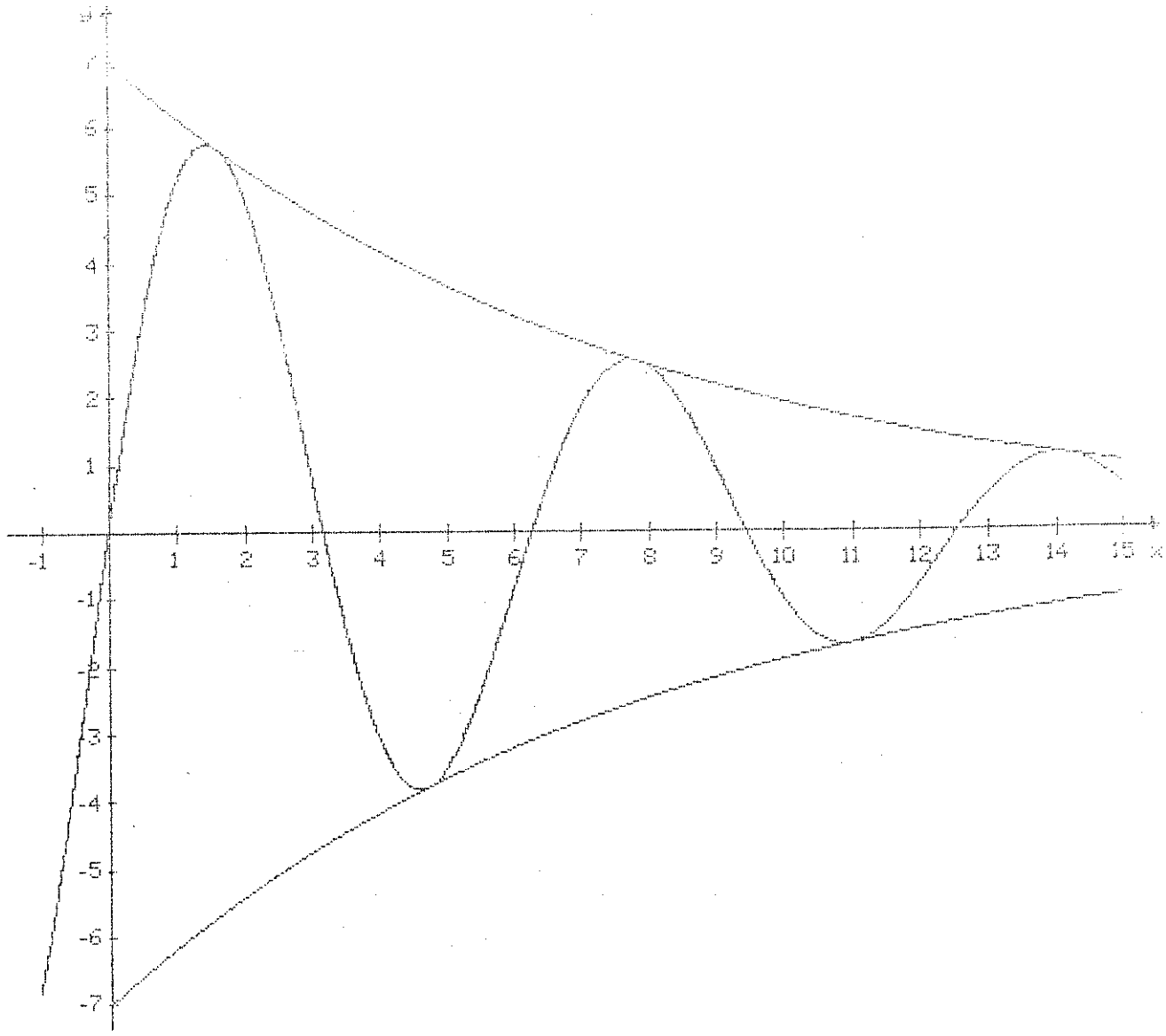
Siegfried Günther, Reutlingen

```
    REM *** Darstellen und Ausdrucken von Funktionsgraphen (BASIC-Teil) ***
    REM 40  70
    REM 30 - 60 nur fuer KEN-DOS-Benutzer
    REM Fuer MODE 8 Zeilen 5800, 6280 und 8100 aendern
    REM Soll MODE 6 bleiben, so sind die x-Bereiche in den Zeilen 5250-5300 um
    REM den Faktor 0,65 zu verkleinern. Ferner ist in Zeile 5550 statt (XR-XL)>16
    REM (XR-XL)>10 einzugeben.
    REM Wird NP12 verwendet,dann sind die Zeilen 5710-5730 und 6010-6090 zu loeschen;
    REM ferner sind alle POKE's in den Zeilen 5760-5790 zu entfernen
    REM
    IF PEEK(#4CF)=1 GOTO 90:REM FLAG, um Programm mehrfach starten zu koennen
    CLEAR #850-#2EC+#100:POKE #29B,#50:POKE #29C,8
    POKE #29D,0:POKE #29E,1:CLEAR #100
    CALLM #F000:REM DLOAD"COPYSYMBOL"
    POKE #4CF,1:REM Symboladressen in Tabelle eintragen
    CALLM #6E0
    GOTO 5000
00  Y!=7*SIN(X!)
10  RETURN
00  REM Zeichnen
10  I=1
20  IF DF!(I)>500 THEN RETURN
30  XB=XB0+INT(DF!(I)*K!):XR!=DF!(I+1):XB1=XB:GOSUB 250:I=I+2:GOTO 220
50  X!=(XB-XB0)/K!
60  IF X!>XR! THEN RETURN
70  GOSUB 100:GOSUB 310:XB1=XB:YB1=YB:FA=F
80  XB=XB+1:GOTO 250
00  REM Funktionsgraph
10  J!=K!*Y!:YB=YB0+INT(J!/2+0.25):F=F2:IF INT(J!+0.5) MOD 2=0 THEN F=F1
15  IF YB>=YT AND YB<=YH THEN DOT XB,YB F
20  REM Luecken werden senkrecht verbunden
25  IF XB1=XB THEN YB1=YB:FA=F:RETURN
30  IF YB=YB1 THEN RETURN
40  IF YB>YB1 GOTO 400
45  REM Kurve faellt
50  IF YB1<YT THEN RETURN
55  IF YB1>YH THEN YB1=YH
60  YB2=YB:IF YB<YT THEN YB2=YT
65  IF YB>YH THEN RETURN
70  IF YB1-YB2>1 THEN DRAW XB,YB2+1 XB,YB1-1 F3
85  IF F=F1 THEN DOT XB,YB2 F3
90  IF FA=F2 THEN DOT XB,YB1 F3
95  RETURN
98  REM Kurve steigt
00  IF YB1<YT THEN YB1=YT
05  IF YB1>YH THEN RETURN
10  YB2=YB:IF YB<YT THEN RETURN
15  IF YB>YH THEN YB2=YH
20  IF YB2-YB1>1 THEN DRAW XB1,YB1+1 XB1,YB2-1 F3
25  IF FA=F1 THEN DOT XB1,YB1 F3
30  IF F=F2 THEN DOT XB1,YB2 F3
40  RETURN
48  REM
49  REM
50  REM Zahlen darstellen
60  R=XB MOD 8:AB=#BFEA-(YMAX-YB)*AY-2*INT(XB/8)
70  POKE #5F0,AB-256*INT(AB/256):POKE #5F1,AB/256:POKE #5F2,R
80  CALLM #600,T$
90  RETURN
98  REM
99  REM
00  REM === Achsenkreuz zeichnen ===
05  AY=PEEK(#9B)
40  DRAW XB0-I+1-K!*ABS(XL),YB0 XB0+15+K!*XR,YB0 F1:REM x-Achse
00  REM Markierungen auf der x-Achse
10  IF XL>=0 GOTO 700
20  FOR I=(-1) TO XL STEP (-1)
30  J=ABS(I)
40  XB=XB0+INT(I*K!+0.5):DRAW XB,YB0-2 XB,YB0+1 F3
50  XB=XB-5:YB=YB0-7:IF I<(-9) THEN XB=XB-2
60  IF I MOD 2<>0 AND K!<23 GOTO 690
80  T$=STR$(I):GOSUB 450
90  NEXT I
```

```
700   FOR I=1 TO XR
710   XB=XB0+INT(I*K!+0.5):DRAW XB,YB0-2 XB,YB0+1 F3
720   XB=XB-2:YB=YB0-7:IF I>9 THEN XB=XB-2
730   IF I MOD 2<>0 AND K!<23 GOTO 770
750   T$=STR$(I):GOSUB 450
770   NEXT I
780   XB=XB0+INT(XR*K!+0.5)+12:YB=YB0-7:T$="x":GOSUB 450
790   YB=YB0-2:T$=">":GOSUB 450
800   REM Markierungen auf der y-Achse
810   IF YU>=0 GOTO 900
820   FOR I=(-1) TO YU STEP (-1)
830   J!=I*K!:YB=YB0+INT(J!/2+0.25):F=F2:IF INT(J!+0.5) MOD 2=0 THEN F=F1
840   DRAW XB0-1,YB XB0+2,YB F
850   YB=YB-1:XB=XB0-13:IF ABS(I)>9 THEN XB=XB-4
860   IF I MOD 2<>0 AND K!<23 GOTO 890
880   T$=STR$(I):GOSUB 450
890   NEXT I
900   FOR I=1 TO YO
910   J!=K!*I:YB=YB0+INT(J!/2+0.25):F=F2:IF INT(J!+0.5) MOD 2=0 THEN F=F1
920   DRAW XB0-1,YB XB0+2,YB F
930   YB=YB-1:XB=XB0-9:IF I>9 THEN XB=XB-4
935   IF I MOD 2<>0 AND K!<23 GOTO 950
940   T$=STR$(I):GOSUB 450
950   NEXT I
955   DRAW XB0,YB0-4-K!*ABS(YU)/2 XB0,YB0+12+K!*YO/2 F3:REM y-Achse
960   REM y und ^
970   XB=XB0-11:YB=INT(YB0+YO*K!/2)+9:T$=" y":GOSUB 450
980   YB=YB+1:XB=XB0-3:T$=" ^":GOSUB 450
990   RETURN
998   REM
999   REM
5000  REM ### Hauptprogramm ###
5010  MODE 0:COLORT 0 13 0 0
5020  F0=0:F1=10:F2=12:F3=13:COLORG F0 F1 F2 F3
5040  ENVELOPE 1 15,10;0,10;
5199  REM
5200  REM == Achsenkreuz ==
5210  PRINT CHR$(12)
5220  CURSOR 2,23:PRINT "Achsenkreuz:"
5230  CURSOR 2,20:PRINT "maximale Aussmasse bei den Laengeneinheiten:";
5240  CURSOR 2,19:PRINT "Laengeneinheit:   x-Wertebereich:   y-Wertebereich:";
5250  CURSOR 2,17:PRINT "0,5 cm ---------->      32                  30";
5260  CURSOR 2,16:PRINT "  1 cm ---------->      16                  15";
5270  CURSOR 2,15:PRINT "  2 cm ---------->       8                   7";
5280  CURSOR 2,14:PRINT "  3 cm ---------->       5                   5";
5290  CURSOR 2,13:PRINT "  4 cm ---------->       4                   3";
5300  CURSOR 2,12:PRINT "  5 cm ---------->       3                   3";
5400  CURSOR 3,10:PRINT "Bitte eingeben:";
5410  CURSOR 3,7:PRINT "Laengeneinheit (nur Zahlen!): ";
5420  CURSOR 3,5:PRINT "x-Achse: von        bis";
5430  CURSOR 3,3:PRINT "y-Achse: von        bis";
5500  XB=33:YB=7:CURSOR XB,YB:INPUT T!:IF T!>500 GOTO 5200
5510  IF T!<0.5 OR T!>5 THEN GOSUB 8500:GOTO 5500
5530  XB=16:YB=5:CURSOR XB,YB:INPUT XL:XB=27:CURSOR XB,YB:INPUT XR
5540  IF XL>500 OR XR>500 GOTO 5200
5550  IF XR<XL OR (XR-XL)*T!>16 THEN GOSUB 8500:XB=16:GOSUB 8600:GOTO 5530
5570  XB=16:YB=3:CURSOR XB,YB:INPUT YU:XB=27:CURSOR XB,YB:INPUT YO
5580  IF YU>500 OR YO>500 GOTO 5200
5590  IF YO<YU OR (YO-YU)*T!>15 THEN GOSUB 8500:XB=16:GOSUB 8600:GOTO 5570
5610  K!=28.33*T!
5710  POKE #300,INT(T!*2)
5720  POKE #304,ABS(XL):IF XL<0 THEN POKE #304,XL+200:POKE #305,XR
5730  POKE #306,ABS(YU):IF YU<0 THEN POKE #306,YU+200:POKE #307,YO
5750  I=14:IF XL>=0 THEN I=25
5760  XB0=INT(I+ABS(XL)*K!):POKE #301,XB0 MOD 256:POKE #302,XB0/256
5780  YB0=INT(7+ABS(YU)*K!/2):POKE #303,YB0
5790  YT=YB0+INT(YU*K!/2)-1:POKE #308,YT:YH=YB0+INT(YO*K!/2)+1:POKE #309,YH
5799  REM
5800  MODE 6:GOSUB 500:REM oder MODE 8
5999  REM
6000  REM == Menue ==
6010  K!=PEEK(#300)*28.33/2:XB0=PEEK(#301)+256*PEEK(#302):YB0=PEEK(#303)
6040  XL=PEEK(#304):IF XL>100 THEN XL=XL-200:XR=PEEK(#305)
6050  YU=PEEK(#306):IF YU>100 THEN YU=YU-200:YO=PEEK(#307)
6060  YT=PEEK(#308):YH=PEEK(#309)
```

```
070  DIM DF!(9):DF!(1)=XL:DF!(2)=XR:DF!(3)=1000
080  F0=0:F1=10:F2=12:F3=13:COLORG F0 F1 F2 F3
090  ENVELOPE 1 15,10;0,10;
100  PRINT CHR$(12)
120  CURSOR 3,3:PRINT "Bild veraendern ==========>B";
130  CURSOR 3,2:PRINT "Ausdrucken ===============>D";
140  T=GETC:IF T=0 GOTO 6140
150  IF CHR$(T)="B" OR CHR$(T)="b" THEN GOSUB 6200
160  IF CHR$(T)="D" OR CHR$(T)="d" THEN GOSUB 8000
190  GOTO 6100
200  REM
210  PRINT CHR$(12)
220  CURSOR 3,3:PRINT "Funktionsterm eingeben ==>F   Menue =>M";
230  CURSOR 3,2:PRINT "Intervalle vorgeben =====>I";
240  CURSOR 3,1:PRINT "Funktionsgraph zeichnen =>Z";
250  CURSOR 3,0:PRINT "Asymptote eingeben ======>A";
260  T=GETC:IF T=0 GOTO 6260
270  IF CHR$(T)="F" OR CHR$(T)="f" THEN GOSUB 6500
280  IF CHR$(T)="Z" OR CHR$(T)="z" THEN MODE 6:GOSUB 200:REM oder MODE 0
290  IF CHR$(T)="I" OR CHR$(T)="i" THEN GOSUB 7000
300  IF CHR$(T)="A" OR CHR$(T)="a" THEN GOSUB 7500
310  IF CHR$(T)="M" OR CHR$(T)="m" THEN RETURN
320  GOTO 6200
498  REM ======
499  REM
500  REM /// Funktionstermeingabe ///
510  PRINT CHR$(12):CURSOR 3,3:PRINT "Eingabe: 100 Y!=f(X!); dann RUN 6000 ";
530  CURSOR 3,2:PRINT "Neue eingeben:";
560  CURSOR 3,1:STOP
600  REM I=100:CALLM #300,I:REM  Wenn lauffaehiges Programm vorhanden
700  REM RETURN
999  REM
000  REM /// Intervalle vorgeben ///
010  PRINT CHR$(12)
020  CURSOR 3,3:PRINT "Intervalle fuer den Graph:";
030  CURSOR 3,2:PRINT "[     ;     ] & [     ;     ]";
040  CURSOR 3,1:PRINT "Wollen Sie z.B. nur das 1.-te, so geben";
050  CURSOR 3,0:PRINT "Sie als 1.-te Zahl im 2.-ten Int. 1000 ein!";
060  XB=4:YB=2:CURSOR XB,YB:INPUT DF!(1):IF DF!(1)<XL OR DF!>XR THEN GOSUB 8500:GOTO 7
60
070  XB=10:CURSOR XB,YB:INPUT DF!(2):IF DF!(2)<DF!(1) OR DF!>XR THEN GOSUB 8500:GOTO 7
70
080  XB=20:CURSOR XB,YB:INPUT DF!(3):IF DF!(3)>500 THEN RETURN
090  IF DF!(3)<DF!(2) OR DF!(3)>XR THEN GOSUB 8500:GOTO 7080
100  XB=26:CURSOR XB,YB:INPUT DF!(4):IF DF!(4)<DF!(3) OR DF!(4)>XR THEN GOSUB 8500:GOT
 7100
110  XB=36:CURSOR XB,YB:INPUT DF!(5):IF DF!(5)>500 THEN RETURN
120  IF DF!(5)<DF!(4) OR DF!(5)>XR THEN GOSUB 8500:GOTO 7110
130  XB=42:CURSOR XB,YB:INPUT DF!(6):IF DF!(6)<DF!(5) OR DF!(6)>XR THEN GOSUB 8500:GOT
 7130
200  XB=50:CURSOR XB,YB:INPUT DF!(7):IF DF!(7)<=500 THEN GOSUB 8500:GOTO 7200
295  RETURN
299  REM
500  REM /// Asymptoten ///
510  PRINT CHR$(12):XB=19:YB=0
520  CURSOR 3,3:PRINT "Hier koennen Sie nur Geraden der Form Y=konst";
530  CURSOR 3,2:PRINT "oder X = konst. eingeben (z.B Y=2):"
540  CURSOR 3,1:PRINT "Wollen Sie keine mehr eingeben, dann 0 eingeben";
550  CURSOR 3,0:PRINT "Bitte eingeben: ";:INPUT T$
560  IF T$="0" THEN RETURN
600  T=LEN(T$):IF T<3 THEN GOSUB 8500:GOTO 7500
610  IF LEFT$(T$,1)<>"Y" AND LEFT$(T$,1)<>"X" THEN GOSUB 8500:GOTO 7500
620  IF MID$(T$,1,1)<>"=" THEN GOSUB 8500:GOTO 7500
640  Z$=RIGHT$(T$,T-2)
700  IF LEFT$(Z$,1)="-" THEN Z$=RIGHT$(Z$,LEN(Z$)-1):T!=(-1)*VAL(Z$):GOTO 7720
710  T!=VAL(Z$)
720  IF LEFT$(Z$,1)<"0" OR LEFT$(Z$,1)>"9" THEN GOSUB 8500:GOTO 7500
730  IF LEFT$(T$,1)="X" AND (T!<XL OR T!>XR) THEN GOSUB 8500:GOTO 7500
740  IF LEFT$(T$,1)="Y" AND (T!<YU OR T!>YO) THEN GOSUB 8500:GOTO 7500
800  REM
810  IF LEFT$(T$,1)="Y" GOTO 7900
820  XB=XB0+INT(T!*K!+0.5):DRAW XB,YB0+YU*K!/2 XB,YB0+YO*K!/2 F3
830  GOTO 7500
900  YB=YB0+INT(T!*K!/2+0.25):F=F2:IF INT(T!*K!+0.5) MOD 2=0 THEN F=F1
910  DRAW XB0+XL*K!,YB XB0+XR*K!,YB F
920  GOTO 7500
```

```
7999  REM
8000  REM /// Ausdruck ///
8010  PRINT CHR$(12):I=13
8020  CURSOR 3,3:PRINT "Drucker einschalten -  dann eine Taste druecken!"
8030  IF I=13 THEN I=12:COLORT 0 I 0 0
8040  IF I=12 THEN I=13:COLORT 0 I 0 0
8050  T=GETC:IF T=0 GOTO 8030
8100  MODE 6:I=#4B0:REM oder MODE 8
8120  T=(XMAX+1)/8:ZBLOX=INT((XR-XL)*K!/8)+6:IF ZBLOX+1>T THEN ZBLOX=T
8130  POKE I+3,ZBLOX
8140  ZBLOY=INT((YO-YU)*K!/8)+8:IF ZBLOY+2>(YMAX+1)/4 THEN ZBLOY=(YMAX+1)/4
8150  POKE I+2,ZBLOY
8160  AY=PEEK(#98):ABILDO=#BFEA-3*AY-((YMAX/4+1-ZBLOY)*4)*AY
8170  POKE I,ABILDO MOD 256:POKE I+1,ABILDO/256
8200  CALLM #4D0
8300  RETURN
8499  REM
8500  REM /// Eingabefehler ///
8510  FOR T=0 TO 4
8520  SOUND 1 0 10 0 FREQ(1000):SOUND 1 0 12 2 FREQ(500)
8530  WAIT TIME 10
8540  NEXT T
8550  SOUND OFF
8600  CURSOR XB,YB:PRINT "     ";:CURSOR XB,YB
8690  RETURN
8999  END
```



L.

PAGE 01 <SYMBOL>

```
002                     * (Symbole im Graphik-Modus darstellen)
003                     *
004                     * Siegfried Guenther, Reutlingen
005                     *
006                     ABILDO  EQU   :5F0      * muss vom BASIC aus
007                     REST    EQU   ABILDO+2  * belegt werden
008                     *
009                     ADRVAR  EQU   REST+1    * Adresse der Textvariablen-
010                     * elemente
011                     ABILDI  EQU   ADRVAR+2  * 'laufende' Bildschirmadr.
012                     YKON    EQU   ABILDI+2
013                     ZSYM    EQU   YKON+2
014                             ORG   ABILDO+:10
015 0600 C5                     PUSH  B
016 0601 E5                     PUSH  H
017 0602 3A9800                 LDA   :98
018 0605 6F                     MOV   L,A
019 0606 2600                   MVI   H,0
020 0608 22F705                 SHLD  YKON
021 060B E1                     POP   H         * Zeiger auf 1. Element
022 060C 5E                     MOV   E,M       * der Textvariablen
023 060D 23                     INX   H         * setzen
024 060E 56                     MOV   D,M
025 060F EB                     XCHG
026 0610 7E                     MOV   A,M       * Zahl der Symbole
027 0611 32F905                 STA   ZSYM      * in der Textvariablen
028 0614 22F305                 SHLD  ADRVAR
029 0617 2AF305         TLOOP   LHLD  ADRVAR
030 061A 23                     INX   H
031 061B 22F305                 SHLD  ADRVAR
032 061E 7E                     MOV   A,M
033 061F FE2E                   CPI   '.'       * . beendet die Ausgabe
034 0621 CA8806                 JE    TEND
035 0624 FE20                   CPI   ' '       * Leerzeichen uebersehen
036 0626 C23606                 JNE   TLAB
037 0629 3AF905                 LDA   ZSYM
038 062C 3D                     DCR   A
039 062D CA8806                 JZ    TEND
040 0630 32F905                 STA   ZSYM
041 0633 C31706                 JMP   TLOOP
042 0636 D628           TLAB    SUI   40        * erstes Symbol ist (
043 0638 6F                     MOV   L,A
044 0639 2600                   MVI   H,0
045 063B 29                     DAD   H         * HL<> = ASTAB
046 063C 110507                 LXI   D,ASTAB   *  +2*(Zeichencode-45)
047 063F 19                     DAD   D
048 0640 5E                     MOV   E,M
049 0641 23                     INX   H
050 0642 56                     MOV   D,M       * DE<> = Adr. des betr. Z.
051 0643 2AF005                 LHLD  ABILDO
052 0646 22F505                 SHLD  ABILDI
053                     *
054 0649 CD8A06                 CALL  SYMBOL
055                     *
056 064C 3AF905                 LDA   ZSYM
057 064F 3D                     DCR   A
058 0650 CA8806                 JE    TEND
059 0653 32F905                 STA   ZSYM
060                     *
061 0656 2AF305                 LHLD  ADRVAR    * Wenn '-' oder '1', dann
062 0659 7E                     MOV   A,M       * kleineren Abstand zum
063 065A FE2D                   CPI   '-'       * naechsten Zeichen
064 065C CA6706                 JE    DICHT
065 065F FE31                   CPI   '1'
066 0661 CA6706                 JE    DICHT
067 0664 C37406                 JMP   NBYTE
068 0667 3AF205         DICHT   LDA   REST
069 066A FE04                   CPI   4
```

PAGE 02 <SYMBOL>

```
070 066C DA7F06                 JLS   ABYTE
071 066F D604                   SUI   4
072 0671 32F205                 STA   REST
073 0674 2AF005         NBYTE   LHLD  ABILDO
074 0677 2B                     DCX   H
075 0678 2B                     DCX   H
076 0679 22F005                 SHLD  ABILDO
077 067C C31706                 JMP   TLOOP
078 067F 2E04           ABYTE   MVI   L,4
079 0681 85                     ADD   L
080 0682 32F205                 STA   REST
081 0685 C31706                 JMP   TLOOP
082                     *
083 0688 C1             TEND    POP   B
084 0689 C9                     RET
085 068A 00             SYMBOL  NOP
086                     *Eine Bildschirmzeile (max. 2 Doppelb.)
087                     *belegen
088                     *
089 068B 0E01           SLOOP   MVI   C,1       * C<>=1:LOW-Byte
090                     *Bildbyte, das durch x und y vorgegeben
091                     *
092 068D 3AF205         SZEILE  LDA   REST
093 0690 47                     MOV   B,A
094 0691 FE00                   CPI   0
095 0693 1A                     LDAX  D
096 0694 CA9D06                 JE    NOSHR
097 0697 B7             SHR     ORA   A         * Wenn Rest<>0 dann
098 0698 1F                     RAR             * verschiebe um Rest<>
099 0699 05                     DCR   B         * nach rechts.
100 069A C29706                 JNZ   SHR
101 069D B6             NOSHR   ORA   M         * Bildbyte =
102 069E 77                     MOV   M,A       * Bildbyte oder Symbolbyte
103                     *'rechts daneben liegendes'
104                     * Bildschirmbyte
105                     *
106 069F 3AF205                 LDA   REST      * Uebertrag auf
107 06A2 47                     MOV   B,A       * das rechte Bild-
108 06A3 FE00                   CPI   0         * schirmbyte?
109 06A5 CAB906                 JE    NCARRY
110 06A8 3E08                   MVI   A,8
111 06AA 90                     SUB   B
112 06AB 47                     MOV   B,A
113 06AC 1A                     LDAX  D
114 06AD B7             SHL     ORA   A
115 06AE 17                     RAL
116 06AF 05                     DCR   B
117 06B0 C2AD06                 JNZ   SHL
118 06B3 2B                     DCX   H
119 06B4 2B                     DCX   H
120 06B5 B6             NOSHL   ORA   M
121 06B6 77                     MOV   M,A
122 06B7 23                     INX   H
123 06B8 23                     INX   H
124 06B9 79             NCARRY  MOV   A,C
125 06BA FE00                   CPI   0
126 06BC CACA06                 JE    SLABEL
127 06BF 0E00                   MVI   C,0
128 06C1 23                     INX   H
129 06C2 13                     INX   D
130 06C3 1A                     LDAX  D
131 06C4 FE01                   CPI   1
132 06C6 C28D06                 JNE   SZEILE
133 06C9 C9                     RET
134 06CA D5             SLABEL  PUSH  D
135 06CB 2AF505                 LHLD  ABILDI
136 06CE EB                     XCHG
137 06CF 2AF705                 LHLD  YKON
```

```
0138 06D2 19                DAD   D
0139 06D3 22F505            SHLD  ABILDI
0140 06D6 D1                POP   D
0141 06D7 13                INX   D
0142 06D8 1A                LDAX  D
0143 06D9 FE01              CPI   1
0144 06DB C28B06            JNE   SLOOP
0145 06DE C9                RET
0146                  * Adressen der Ziffern in die Tabelle
0147                  * ATAB eintragen
0148                        ORG   :6E0
0149 06E0 C5                PUSH  B
0150 06E1 01AF07            LXI   B,STAB
0151 06E4 110507            LXI   D,ASTAB
0152 06E7 0A          TALOOP LDAX  B
0153 06E8 FE7B              CPI   123       * 123 steht fuer
0154 06EA CA0307            JE    TAEND     * letztes Zeichen
0155                  *
0156 06ED D628              SUI   40        * Tab. beginnt ab '('
0157 06EF 6F                MOV   L,A
0158 06F0 2600              MVI   H,0       * HL<> = ASTAB
0159 06F2 29                DAD   H         *       + (ASC(Symbol)-40)*2
0160 06F3 19                DAD   D
0161                  *
0162 06F4 03                INX   B         * Symboladr. auf 1.Bildbyte
0163 06F5 71                MOV   M,C       * diese Adr. unter ASTAB
0164 06F6 23                INX   H         * abspeichern
0165 06F7 70                MOV   M,B
0166                  *
0167 06F8 03          SLAENG INX   B         * Symboladresse auf
0168 06F9 0A                LDAX  B         * letztes Byte des
0169 06FA FE01              CPI   1         * Symbols bringen (= 1)
0170 06FC C2F806            JNE   SLAENG
0171                  *
0172 06FF 03                INX   B         * Symboladr. auf naechstes
0173 0700 C3E706            JMP   TALOOP    * Symbol
0174                  *
0175 0703 C1          TAEND  POP   B
0176 0704 C9                RET
0177                  *
0178 0705 00          ASTAB  NOP             * Platz fuer 86
0179                  * Adresseneintraege
0180                        ORG   ASTAB+170
0181 07AF 2D0000      STAB   DATA  '-',0,0,224,1
0182 07B4 307088            DATA  '0',112,136,136,136,136,136,112,1
0183 07BD 31E040            DATA  '1',224,64,64,64,64,192,64,1
0184 07C6 32F880            DATA  '2',248,128,64,48,8,136,112,1
0185 07CF 337088            DATA  '3',112,136,8,48,16,8,248,1
0186 07D8 341010            DATA  '4',16,16,248,144,80,48,16,1
0187 07E1 357088            DATA  '5',112,136,8,8,240,128,248,1
0188 07EA 367088            DATA  '6',112,136,136,240,128,64,56,1
0189 07F3 374040            DATA  '7',64,64,64,32,16,8,248,1
0190 07FC 387088            DATA  '8',112,136,136,112,136,136,112,1
0191 0805 39E010            DATA  '9',224,16,8,120,136,136,112,1
0192 080E 78B850            DATA  'x',136,80,32,80,136,1
0193 0815 79F008            DATA  'y',240,8,120,136,136,136,136,1
0194 081E 3E00B0            DATA  '>',0,128,192,224,240,224,192,128,1
0195 0828 5EFE7C            DATA  '^',254,124,56,16,1
0196 082E 7B01              DATA  123,1
0197 0830                   END

****************************
* S Y M B O L   T A B L E *
****************************

ABILDO 05F0   ABILDI 05F5   ABYTE  067F   ADRVAR 05F3
ASTAB  0705   DICHT  0667   NBYTE  0674   NCARRY 06B9
NOSHL  06B5   NOSHR  069D   REST   05F2   SHL    06AD

PAGE 04    <SYMBOL>

SHR    0697   SLABEL 06CA   SLAENG 06F8   SLOOP  068B
STAB   07AF   SYMBOL 068A   SZEILE 068D   TAEND  0703
TALOOP 06E7   TEND   0688   TLAB   0636   TLOOP  0617
YKON   05F7   ZSYM   05F9
```



```
P.
0600 C5 E5 3A 98 00 6F 26 00 22 F7 05 E1 5E 23 56 EB
0610 7E 32 F9 05 22 F3 05 2A F3 05 23 22 F3 05 7E FE
0620 2E CA 88 06 FE 20 C2 36 06 3A F9 05 3D CA 88 06
0630 32 F9 05 C3 17 06 D6 28 6F 26 00 29 11 05 07 19
0640 5E 23 56 2A F0 05 22 F5 05 CD BA 06 3A F9 05 3D
0650 CA 88 06 32 F9 05 2A F3 05 7E FE 2D CA 67 06 FE
0660 31 CA 67 06 C3 74 06 3A F2 05 FE 04 DA 7F 06 D6
0670 04 32 F2 05 2A F0 05 2B 2B 22 F0 05 C3 17 06

067F 2E
0680 04 85 32 F2 05 C3 17 06 C1 C9 00 0E 01 3A F2 05
0690 47 FE 00 1A CA 9D 06 B7 1F 05 C2 97 06 B6 77 3A
06A0 F2 05 47 FE 00 CA B9 06 3E 08 90 47 1A B7 17 05
06B0 C2 AD 06 2B 2B B6 77 23 23 79 FE 00 CA CA 06 0E
06C0 00 23 13 1A FE 01 C2 BD 06 C9 D5 2A F5 05 EB 2A
06D0 F7 05 19 22 F5 05 D1 13 1A FE 01 C2 8B 06 C9

06E0 C5 01 AF 07 11 05 07 0A FE 7B CA 03 07 D6 28 6F
06F0 26 00 29 19 03 71 23 70 03 0A FE 01 C2 F8 06 03
0700 C3 E7 06 C1 C9 00

07AF 2D
07B0 00 00 E0 01 30 70 88 88 88 88 88 70 01 31 E0 40
07C0 40 40 40 C0 40 01 32 F8 80 40 30 08 88 70 01 33
07D0 70 88 08 30 10 08 F8 01 34 10 10 F8 90 50 30 10
07E0 01 35 70 88 08 08 F0 80 F8 01 36 70 88 88 F0 80
07F0 40 38 01 37 40 40 40 20 10 08 F8 01 38 70 88 88
0800 70 88 88 70 01 39 E0 10 08 78 88 88 70 01 78 88
0810 50 20 50 88 01 79 F0 08 78 88 88 88 88 01 3E 00
0820 80 C0 E0 F0 E0 C0 80 01 5E FE 7C 38 10 01

082E 7B 01
```

L.

PAGE 01 COPY (Hardcopy fuer Funktionsgraphen)

```
002                     * Siegfried Guenther, Reutlingen
003                     *
004                     VARIAB  EQU   :4B0
005                     ABILD0  EQU   VARIAB
006                     ZBLOY   EQU   ABILD0+2
007                     ZBLOX   EQU   ZBLOY+1
008                     ZBLOXI  EQU   ZBLOX+1
009                     ABILD1  EQU   ZBLOXI+1
010                     ABILDI  EQU   ABILD1+2
011                     YKON1   EQU   ABILDI+2
012                     YKON2   EQU   YKON1+2
013                     XWERT   EQU   YKON2+2
014                     PUFFA   EQU   XWERT+4
015                     PUFFE   EQU   PUFFA+7
016                     START   EQU   VARIAB+:20
017                     *
018                             ORG   START
019                     *
020 04D0 F5                     PUSH  PSW
021 04D1 C5                     PUSH  B
022 04D2 D5                     PUSH  D
023 04D3 E5                     PUSH  H
024 04D4 3A9B00                 LDA   :98       * Adressenabstand zweier
025 04D7 6F                     MOV   L,A       * Bilschirmzeilen in
026 04D8 2600                   MVI   H,0       * YKON1
027 04DA 22B904                 SHLD  YKON1
028 04DD 29                     DAD   H         * YKON2<> = 4 YKON1<>
029 04DE 29                     DAD   H
030 04DF 22BB04                 SHLD  YKON2
031 04E2 3AB304                 LDA   ZBLOX     * XWERT<> = 8 ZBLOX<>
032 04E5 6F                     MOV   L,A
033 04E6 2600                   MVI   H,0
034 04E8 29                     DAD   H
035 04E9 29                     DAD   H
036 04EA 29                     DAD   H
037 04EB 22BD04                 SHLD  XWERT
038                     *
039 04EE 3EA9                   MVI   A,:A9
040 04F0 3203FE                 STA   :FE03
041 04F3 0E1B                   MVI   C,27      * Drucker normieren
042 04F5 CDA705                 CALL  PRINT
043 04FB 0E40                   MVI   C,64
044 04FA CDA705                 CALL  PRINT
045 04FD 0E1B                   MVI   C,27      * Drucker auf
046 04FF CDA705                 CALL  PRINT     * Zeilenabstand
047 0502 0E33                   MVI   C,'3'     * 24/216 '' setzen
048 0504 CDA705                 CALL  PRINT
049 0507 0E18                   MVI   C,24
050 0509 CDA705                 CALL  PRINT
051                     *
052 050C 2AB004         ZEILEL  LHLD  ABILD0    * $ neue Viererzeile
053 050F 22B504                 SHLD  ABILD1    * $
054 0512 22B704                 SHLD  ABILDI    * $
055 0515 3AB304                 LDA   ZBLOX     * $
056 0518 32B404                 STA   ZBLOXI    * $
057 051B 0E1B                   MVI   C,27      * $ Drucker in Graphic-
058 051D CDA705                 CALL  PRINT     * $ Modus setzen
059 0520 0E2A                   MVI   C,'*'     * $
060 0522 CDA705                 CALL  PRINT     * $
061 0525 0E05                   MVI   C,5       * $ Plott-Modus
062 0527 CDA705                 CALL  PRINT     * $
063 052A 3ABD04                 LDA   XWERT     * $
064 052D 4F                     MOV   C,A       * $
065 052E CDA705                 CALL  PRINT     * $
066 0531 3ABE04                 LDA   XWERT+1   * $
067 0534 4F                     MOV   C,A       * $
068 0535 CDA705                 CALL  PRINT     * $
069 0538 0E01           YLOOP   MVI   C,1       * /
```

PAGE 02 COPY (Hardcopy fuer Funktionsgraphen)

```
070                     *
071 053A CDB405         BLOCLO  CALL  BYTE      * %
072 053D 23                     INX   H         * %
073 053E 79                     MOV   A,C       * %
074 053F B7                     ORA   A         * %
075 0540 17                     RAL             * %
076 0541 4F                     MOV   C,A       * %
077 0542 CDB405                 CALL  BYTE      * %
078 0545 2AB704                 LHLD  ABILDI    * %
079 0548 EB                     XCHG            * %
080 0549 2AB904                 LHLD  YKON1     * %
081 054C 19                     DAD   D         * %
082 054D 22B704                 SHLD  ABILDI    * %
083 0550 79                     MOV   A,C       * %
084 0551 B7                     ORA   A         * %
085 0552 17                     RAL             * %
086 0553 4F                     MOV   C,A       * %
087 0554 D23A05                 JNC   BLOCLO    * %
088                     *
089 0557 21C104                 LXI   H,PUFFA   * &
090 055A 0608                   MVI   B,8       * & 8 Byte's ausdrucken
091 055C 3E00           PLOOP   MVI   A,0       * &
092 055E 4E                     MOV   C,M       * &
093 055F 77                     MOV   M,A       * & Pufferbyte auf 0
094 0560 CDA705                 CALL  PRINT     * &
095 0563 05                     DCR   B         * &
096 0564 23                     INX   H         * &
097 0565 C25C05                 JNZ   PLOOP     * &
098                     *
099 0568 2AB504                 LHLD  ABILD1    * / Adresse auf die
100 056B 2B                     DCX   H         * / naechsten 8 Spalten
101 056C 2B                     DCX   H         * / der 4er-Bildzeile
102 056D 22B504                 SHLD  ABILD1    * / bringen
103 0570 22B704                 SHLD  ABILDI    * /
104 0573 3AB404                 LDA   ZBLOXI    * / Wenn ZBLOXI = 0,
105 0576 3D                     DCR   A         * / dann ist die 4er-
106 0577 32B404                 STA   ZBLOXI    * / Bildzeile fertig
107 057A C23805                 JNZ   YLOOP     * /
108                     *
109 057D 0E0D                   MVI   C,:0D     * $ Gebe Zeilenende
110 057F CDA705                 CALL  PRINT     * $ zum Drucker
111 0582 0E0A                   MVI   C,10      * $
112 0584 CDA705                 CALL  PRINT     * $
113 0587 2ABB04                 LHLD  YKON2     * $ Berechne Adresse
114 058A 3AB004                 LDA   ABILD0    * $ der neuen
115 058D 95                     SUB   L         * $ Vierer-Bildzeile
116 058E 32B004                 STA   ABILD0    * $
117 0591 3AB104                 LDA   ABILD0+1  * $
118 0594 9C                     SBB   H         * $
119 0595 32B104                 STA   ABILD0+1  * $
120 059B 3AB204                 LDA   ZBLOY     * $ Wenn noch nicht
121 059B 3D                     DCR   A         * $ alle 4er-Zeilen
122 059C 32B204                 STA   ZBLOY     * $ fertig, dann
123 059F C20C05                 JNZ   ZEILEL    * $ weiter
124                     *
125                     *
126 05A2 E1                     POP   H         * Ende
127 05A3 D1                     POP   D
128 05A4 C1                     POP   B
129 05A5 F1                     POP   PSW
130 05A6 C9                     RET
131                     *
132 05A7 3A02FE         PRINT   LDA   :FE02
133 05AA E610                   ANI   :10
134 05AC C2A705                 JNE   PRINT
135 05AF 79                     MOV   A,C
136 05B0 3200FE                 STA   :FE00
137 05B3 C9                     RET
```

```
PAGE 03    COPY (Hardcopy fuer Funktionsgraphen)

138                         *
139 05B4 11C804             BYTE    LXI   D,PUFFE
140 05B7 0601                       MVI   B,1
141 05B9 7E                 BYTELO  MOV   A,M
142 05BA A0                         ANA   B
143 05BB CAC105                     JZ    BYTEL1
144 05BE 1A                         LDAX  D
145 05BF B1                         ORA   C
146 05C0 12                         STAX  D
147 05C1 1B                 BYTEL1  DCX   D
148 05C2 78                         MOV   A,B
149 05C3 B7                         ORA   A
150 05C4 17                         RAL
151 05C5 47                         MOV   B,A
152 05C6 D2B905                     JNC   BYTELO
153 05C9 C9                         RET
154 05CA                            END

*****************************
* S Y M B O L   T A B L E *
*****************************

ABILDO 04B0   ABILD1 04B5   ABILDI 04B7   BLOCLO 053A
BYTE   05B4   BYTEL1 05C1   BYTELO 05B9   FLOOP  055C
PRINT  05A7   PUFFA  04C1   PUFFE  04C8   START  04D0
VARIAB 04B0   XWERT  04BD   YKON1  04B9   YKON2  04BB
YLOOP  0538   ZBLOX  04B3   ZBLOXI 04B4   ZBLOY  04B2
ZEILEL 050C

P.
04D0 F5 C5 D5 E5 3A 98 00 6F 26 00 22 B9 04 29 29 22
04E0 BB 04 3A B3 04 6F 26 00 29 29 29 22 BD 04 3E A9
04F0 32 03 FE 0E 1B CD A7 05 0E 40 CD A7 05 0E 1B CD
0500 A7 05 0E 33 CD A7 05 0E 18 CD A7 05 2A B0 04 22
0510 B5 04 22 B7 04 3A B3 04 32 B4 04 0E 1B CD A7 05
0520 0E 2A CD A7 05 0E 05 CD A7 05 3A BD 04 4F CD A7
0530 05 3A BE 04 4F CD A7 05 0E 01 CD B4 05 23 79 B7
0540 17 4F CD B4 05 2A B7 04 EB 2A B9 04 19 22 B7 04

0550 79 B7 17 4F D2 3A 05 21 C1 04 06 08 3E 00 4E 77
0560 CD A7 05 05 23 C2 5C 05 2A B5 04 2B 2B 22 B5 04
0570 22 B7 04 3A B4 04 3D 32 B4 04 C2 38 05 0E 0D CD
0580 A7 05 0E 0A CD A7 05 2A BB 04 3A B0 04 95 32 B0
0590 04 3A B1 04 9C 32 B1 04 3A B2 04 3D 32 B2 04 C2
05A0 0C 05 E1 D1 C1 F1 C9 3A 02 FE E6 10 C2 A7 05 79
05B0 32 00 FE C9 11 C8 04 06 01 7E A0 CA C1 05 1A B1
05C0 12 1B 78 B7 17 47 D2 B9 05 C9
```

Vor dem Abspeichern Speicherinhalt von #4B0 – #4CF auf 0 setzen (z.B. mit MT und S); sonst wird der leicht gefüllte Puffer mit ausgedruckt!

Ultrahochauflösende Grafik auf Ihrem Drucker

von Gordon Wassermann

Ich möchte Ihnen hier ein Basicprogramm vorstellen, das in der Lage ist, Graphen von reellen Funktionen mit sehr hoher Auflösung auf einem beliebigen grafikfähigen Drucker zu plotten. Bei modernen Druckern ist das Ergebnis wesentlich besser, als auf dem Bildschirm, oder entsprechend bei einem Hardcopy vom Bildschirm, möglich ist. Einzelne Nadelpunkte sind auf der gedruckten Kurve nicht mehr sichtbar und die Sprünge zwischen den Geradenstücken, aus denen die Kurve ja immer noch besteht, fallen kaum noch auf--die Grafik sieht praktisch wie von einem Plotter gezeichnet aus. Bei meinem Drucker, dem Epson FX-80, beträgt die Auflösung 1920 Punkte in der Waagerechten (oder 240 Punkte pro Zoll) mal 1440 Punkte in der Senkrechten (oder effektiv 144 Punkte pro Zoll--es wird jeweils ein Zoll am oberen und unteren Papierrand freigelassen). Das macht insgesamt 2764800 Bildpunkte, mehr als das 21fache der maximalen Bildschirmauflösung!

Ich wurde genötigt, das Programm zu schreiben, weil ich meinen Ingenieurstudenten an der Universität einige mathematische Funktionen vorführen wollte, und dafür gute projektionsfähige Bilder brauchte. Aber zusätzlich dazu ermuntert wurde ich durch einen Artikel in der Novemberausgabe (1985) der amerikanischen Computerzeitschrift *BYTE*, in dem genau ein solches Programm vorgestellt und diskutiert wurde. Nur konnte ich das Programm aus *BYTE* leider nicht übernehmen, zum einen weil es in Turbo Pascal für den IBM PC geschrieben war, aber wichtiger, weil es das gesamte Druckerbild zuerst einmal im Rechnerspeicher zeichnete und dafür einen Puffer von 128 Kilobyte benötigte! So viel Speicher hat der DAI bekanntlich nicht. Außerdem erzielte das *BYTE* Programm nur eine Auflösung von 1600 x 640 Punkten, wegen Einschränkungen für die Größe von Arrays in Pascal! Deshalb habe ich für den DAI ein ganz neues Programm geschrieben. Ich habe dennoch aus dem *BYTE* Artikel nützliche Anregungen beziehen können, insbesondere was die Handhabung des Druckers betrifft, auch wenn die Empfehlungen der *BYTE* Autoren sich nicht in allen Punkten als günstig erwiesen haben (s. unten).

Die hohe Auflösung wird erzielt, in dem man zum einen die Grafik mit dem dichtesten Grafikmodus des Druckers ausgibt, d.h. mit der höchsten waagerechten Auflösung, die der Drucker drucken kann, und zum anderen, in dem man in der Senkrechten den feinsten Zeilenvorschub, den der Drucker beherrscht, ausnutzt, um auch senkrecht Punkte dichter zu setzen als der Nadelabstand, etwa mit halbem Nadelabstand. Durch diese Dichte in beiden Richtungen wird erreicht, daß benachbarte Punkte verschmelzen und die gezeichnete Kurve kontinuierlich aussieht, und es nicht mehr auffällt, daß sie aus einzelnen Punkten besteht.

Dieses Verfahren wird von dem am Schluß dieses Artikels wiedergegebenen Basicprogramm verwirklicht, welches ich zum besseren Verständnis seiner Wirkungsweise mit viel Kommentar zwischen den Zeilen versehen habe.

Um in den typischsten Fällen das Papier möglichst gut auszufüllen wird der Graph im Querformat gezeichnet. D.h., die y-Richtung verläuft über die Breite des Papiers, und die x-Richtung in Längsrichtung, also parallel zum gelochten Randstreifen bei Endlospapier.

Der Bildausschnitt kann frei gewählt werden. Es schadet auch nichts, wenn die Kurve aus dem Bildausschnitt herausläuft--es werden dann eben genau die Teile der Kurve gezeichnet, die im Bildausschnitt sichtbar sind.

Bei einer so präzisen Zeichnung, wie dieses Verfahren verspricht, will man ja auch die genaue Lage der Kurve in der Ebene sichtbar haben. Deshalb werden die Koordinatenachsen auch mitgezeichnet, wenn sie im Bildausschnitt sichtbar sind, und sie können auch mit einer Maßstabsmarkierung versehen werden. Eine Beschriftung der Zeichnung ist allerdings nicht vorgesehen. Zum einen müßte dazu extra ein Zeichensatz erstellt werden, da der normale Druckerzeichensatz nicht zum Querformat paßt und um 90° gedreht werden müßte. Diesen Zeichensatz würde man sich vielleicht sogar in der gleichen hohen Auflösung wünschen (obwohl das eigentlich nicht unbedingt erforderlich wäre). Zum anderen wäre es aber auch sehr umständlich, im Programm zu gewährleisten, daß die Schrift nicht von der zu plottenden Kurve durchkreuzt wird. Daher habe ich auf eine Beschriftung verzichtet. Notfalls kann man die Bilder ja auch mit der Schreibmaschine nachträglich beschriften, was ich bei meinen Vorführbildern für die Ingenieure auch getan habe.

Am Schluß dieses Artikels finden Sie zwei Probeausdrücke auf einem Epson FX-80--das erste Bild zeigt die Funktion $\tanh x = (\exp(x) + \exp(-x))/2$ im Bereich zwischen $x = -3$ und $x = 3$. Das Zweite zeigt die Funktion $\sqrt{|x|}\sin(\frac{1}{x})$ im Bereich $x = -1$ bis $x = 1$.

Zur Anwendung des Programms: vor dem Start muß man in die Zeilen 3200-3300 die Berechnung der zu plottenden Funktion hineinschreiben. Genau wie das zu geschehen hat, wird in den Kommentarzeilen zum Programm an der Stelle hinreichend erklärt. Die hier wiedergegebene Version des Programms berechnet die Funktion $\tanh x = (\exp(x) + \exp(-x))/2$.

Nach der Eingabe von **RUN** fragt das Programm dann zuerst nach dem gewünschten Bildausschnitt, d.h. nach dem zu zeichnenden Bereich der x-Werte und der y-Werte. Man kann hier alle Werte explizit eingeben, und das Programm skaliert das Bild dann in beiden Richtungen nach den eingegebenen Werten. Man kann aber auch für **eine** der beiden Achsen die Bestimmung von einem oder von beiden Endpunkten des zu zeichnenden Intervalls dem Programm überlassen--dazu gebe man einfach 999 ein, wenn nach diesem Endpunkt gefragt wird. In diesem Fall verwendet das Programm für diese Achse das gleiche Längenmaß auf dem Papier, wie für die andere Achse (das Bild wird also maßstabsgleich in beiden Richtungen), und wenn ein Endpunkt explizit angegeben wurde, wird der Andere aus Diesem gemäß diesem Maßsstab errechnet. Wurden beide Endpunkte unbestimmt gelassen, wird der Maßstab der anderen Achse übernommen und das Bild wird symmetrisch um diese Achse ausgerichtet.

Anschließend fragt das Programm, wie die Skalierung auf die Achsen zu markieren ist. Es ist eine zweistufige Markierung vorgesehen, mit langen Markierungsstrichen für die Einteilung in Haupteinheiten und kürzeren Strichen für eine zusätzliche feinere Unterteilung dieser Hauptintervalle. Das Programm fragt zuerst nach der Länge der Hauptintervalle. Falls keine Markierung gewünscht wird, kann hier 999 oder eine negative Zahl eingegeben werden. Anschließend wird danach gefragt, in wieviele feinere Einheiten jedes Hauptintervall unterteilt werden soll. Wenn man nur eine einstufige Markierung wünscht, kann man hier 1 eingeben. Die Abfrage über die Markierung wird für jede Achse getrennt vorgenommen, so daß man z.B. auch nur eine Achse markieren kann, wenn man will.

Nach dieser Abfrage folgt eine ziemlich lange Pause, während das Programm die zu plottenden Funktionswerte berechnet. Spätestens während dieser Zeit sollte man den Drucker einschalten. Nach ein paar Minuten beginnt er dann zu drucken, und es dauert nicht lange, bis das Bild fertig ist.

Alle druckerspezifische Daten sind in nur drei oder vier Programmzeilen konzentriert (z.B. in einigen Variabelzuweisungen ganz zu Beginn), damit das Programm, hier für den Epson FX-80 geschrieben, sich mühelos auf beliebige andere Druckertypen anpassen läßt (nur grafikfähig muß der Drucker schon sein, und die Qualität der Ergebnisse hängt natürlich auch von den Fähigkeiten des Druckers ab).

Anpassung des Programms an ihr Drucker:

Für den Epson FX-80 oder Kompatible ist keine Anpassung nötig. Für andere Drucker sind nur wenige Änderungen nötig, wie im folgenden beschrieben:

In Zeile 20 setze man **HZOLL!** und **VZOLL!** auf die tatsächlichen Maße, waagerecht bzw. senkrecht, der zu bedruckenden Papierfläche (gemeint sind nicht die Maße des **Bogens**, sondern wirklich nur des Druckbildes!). Ich habe die Maße auf Zoll angegeben, aber Sie können die Maße auch in einem beliebigen anderen Längenmaß angeben, weil nur das Verhältnis (**RATIO!**) der beiden Dimensionen später verwendet wird. Man beachte, daß Endlospapier in Deutschland normalerweise 8 x 12 Zoll mißt, aber da der Druckkopf des Epson FX-80 beim Blattanfang gewöhnlich ein Zoll unterhalb des Blattrands steht, fand ich es bequemer, oben und unten einen Zoll freizulassen und nur 10 Zoll zu bedrucken.

In Zeile 30 setze man **HRES** auf die Auflösung, in Punkten pro Zeile, des feinstauflösenden Bitgrafikmodus (in der Waagerechten), den Ihr Drucker besitzt (beim Epson MX-80 z.B. 960 Punkte, beim Itoh 8510A z.B. 1280 Punkte). **VRES** setze man auf die senkrechte Auflösung, in Punkten pro Spalte der **Druckfläche** (nicht des Blatts!), die erreicht werden kann unter Ausnutzung von Zeilenvorschüben von einer halben Punkthöhe, wenn Ihr Drucker dazu fähig ist. Da eine Punkthöhe gerade 1/72 Zoll beträgt, ist diese senkrechte Auflösung 144 x Höhe der Druckfläche in Zoll. Man beachte: viele Drucker, wie der Itoh 8510A, bieten wirklich Zeilenvorschübe von 1/2 Punkt oder 1/144 Zoll. Die Epson Drucker, wie der FX-80 z.B., können aber nur Zeilenvorschübe von 1/3 Punkt oder 1/216 Zoll! Es ist aber trotzdem besser, so zu tun, als wäre dies ein Zeilenvorschub von 1/144 Zoll, d.h., zwischen zwei Punktreihen nur eine weitere Punktreihe zu drucken und nicht zwei, weil das Bild sonst zu schwarz wird und feine Details eher verwischt werden als daß sie zur Geltung kämen. Der 1/3-punktige Zeilenvorschub ist ohnehin nicht ganz genau, so daß der Unterschied zu einem echten halbpunktigen Zeilenvorschub nicht sichtbar ist.

In Zeile 60 müssen Sie in den ersten **Bytes** des Arrays **BUFF** (der nur als Druckpuffer dient und also nicht wirklich als Integerarray verwendet wird) die Steuerfolge speichern, die gerade Ihren Drucker in den höchstauflösenden Grafikmodus versetzt und die Übersendung einer vollen Zeile Grafikbitmusterdaten ankündigt--hinter dieser Steuerfolge wird das Plotprogramm die Bitmusterdaten selber ablegen. Setzen Sie in Zeile 40 die Variable **PREF** auf die Länge dieser Steuerfolge (in Bytes); aus **PREF** und **HRES** wird die richtige Dimensionierung von **BUFF** errechnet, und aus **PREF** errechnet das Programm auch die Adresse, wo die eigentlichen Bitmusterdaten beginnen im Puffer.

Die Bitmusterdaten werden so im Puffer abgelegt, daß die höherwertigen Bits in jedem Grafikbyte den höherliegenden Bildpunktreihen auf dem Papier entsprechen--so erwarten die Epson Drucker ihre Bitgrafikdaten. Falls Ihr Drucker, wie etwa der Itoh 8510A, die Bitgrafikdaten gerade in der umgekehrten Reihenfolge erwartet (höherwertige Bits eines Grafikbytes entsprechen tieferliegenden Punkten auf dem Papier), müssen Sie in Zeile 5100 **MASKE=#100** durch **MASKE=1** ersetzen und den Befehl **MASKE=MASKE SHR 1** entfernen, dafür aber in Zeile 5600 einen Befehl **MASKE=MASKE SHL 1** vor **NEXT J** einfügen.

Jetzt wieder für alle Drucker (außer FX-80): Zeile 5800 müssen Sie ersetzen durch einen Printbefehl, der auf Ihrem Drucker den Zeilenvorschub für K=0 auf 1/2 Punkt und für K=1 auf 7 1/2 Punkte setzt (bei Epson Druckern, 1/3 Punkt bzw. 7 2/3 Punkte) und dann einen Zeilenvorschub **mit Wagenrücklauf** ausführt (bei Epson Druckern reicht schon der Zeilenvorschub allein, um auch einen Wagenrücklauf zu bewirken!).

Und schließlich, das **PRINT** in Zeile 6100 muß so geändert werden, daß es auf Ihrem Drucker den Zeilenvorschub wieder auf die normalen 6 Zeilen pro Zoll stellt, eventuell wenn nötig den Grafikmodus ausschaltet und einen normalen Textmodus einschaltet, und schließlich einen Blattvorschub macht.

Mit diesen Änderungen können Sie das Programm dann mit Ihrem Drucker verwenden.

Falls Ihr Drucker nur in Vielfachen von einer ganzen Punkthöhe Zeilenvorschübe machen kann, sind weitergehende Änderungen des Programms nötig, auf die ich hier nicht in allen Details eingehen möchte. Insbesondere müssen ab Zeile 1000 die Zahlen 16 und 15 wo immer sie erscheinen (z.B. bei der Definition von **ZAEHLER!** und bei der zweiten Dimension der Arrays) durch 8 bzw. 7 ersetzt werden. Die **FOR**-Schleife über K zwischen Zeilen 4800 und 5900 kann entfallen (in anderen Worten, der Inhalt der Schleife muß dann nur einmal durchlaufen werden), und der **FOR** Befehl in Zeile 5100 kann jetzt **FOR J=0 TO 7** (STEP 1) lauten. In Zeile 5800 ist einfach ein Zeilenvorschub von 8 Punkthöhen (genau eine volle Grafikzeile) zu machen.

Zum Schluß noch einige kurze Bemerkungen über einige Besonderheiten der Druckerhandhabung im Falle des Epson FX-80.

Bei dem hier verwendeten höchstauflösenden Grafikmodus mit vierfacher Dichte (1920 Punkte pro Zeile) kann der Drucker nicht zwei horizontal nebeneinanderliegende Punkte drucken--zwischen zwei gedruckten Punkten muß also immer mindestens ein Punkt frei bleiben. Wird von den Bitmusterdaten trotzdem das Drucken von benachbarten Punkten verlangt, so wird nur der erste Punkt gedruckt und der zweite wird einfach übergangen. Man könnte meinen, daß dadurch ein Teil der Auflösung verloren geht, aber es stellt sich heraus, daß diese Lücken in durchgehenden Geraden (z.B. in der y-Achse) nicht sichtbar sind, und die Punkte, die tatsächlich gedruckt werden werden ja auch mit der geforderten Genauigkeit auf's Papier gesetzt. Dieses Auslassen von Punkten fällt überhaupt nur auf, und dann in sehr geringem Maße, wenn in einer *senkrecht* verlaufenden Gerade (z.B. die *x*-Achse) einige Punkte nicht gedruckt werden, weil in manchen Druckzeilen die direkt links danebenliegenden Punkte gesetzt sind--dann erhält die senkrechte Gerade einen kaum merkbaren kleinen Knick.

Ich habe es ausprobiert, die dichte Auflösung hundertprozentig zu erreichen, indem ich jede Zeile zweimal habe drucken lassen, einmal nur die Punkte an geraden Stellen, beim zweiten Mal nur die Punkte an den ungeraden Stellen, aber ich mußte feststellen, daß dadurch die Druckqualität keine nennenswerte Besserung erfuhr, ja sich sogar verschlechterte, weil durch den überhohen Schwärzungsgrad feine Details zugekleistert wurden.

Ähnlich verhält es sich, wie schon erwähnt, mit der senkrechten Auflösung. Eigentlich könnte der Epson Drucker zwischen je zwei Punktreihen mit seinem Zeilenvorschub von 1/3 Punkt sogar zwei zusätzliche Punktreihen setzen, und nicht nur eine, wie wir's gemacht haben, aber ich meine, dies würde ohne sichtbare Verbesserung des Bildes auch nur feine Bilddetails zuschwärzen. Die Zwischenreihen ganz wegzulassen (also überhaupt nicht mit halb- oder drittelpunktigem Zeilenvorschub zu arbeiten) empfiehlt sich aber nicht--ich habe es probehalber gemacht, und die Minderung der senkrechten Auflösung war deutlich sichtbar.

Natürlich kann man sich vorstellen, daß bei den Epson Druckern, die keinen echten halbpunktigen Zeilenvorschub haben, vielleicht eine andere Kombination als 1/3 Punkt und 7 2/3 Punkte günstiger wäre oder vielleicht bessere Ergebnisse liefert. Die Autoren des *BYTE* Artikels haben verschiedene Mäglichkeiten ausprobiert, und sie empfiehlen sogar die Kombination 1/3 Punkt und 7 1/3 Punkte! Ich habe dies aber dann auch versucht und dabei eine deutliche Verkürzung des Bildes in der Längsrichtung festgestellt, so daß ich diese Kombination doch nicht akzeptabel fand, zumal die Ergebnisse bei 1/3 und 7 2/3 einwandfrei sind. Vielleicht arbeitet mein FX-80 etwas präziser als der von den *BYTE* Autoren! (So abwegig ist das nicht--ein Bekannter von mir, der einen der ersten FX-80 gekauft hat, lobt das Schriftbild von meinem Drucker sehr in Vergleich zu seinem).

Wie dem auch sei, jedem steht es frei zu experimentieren!

Hier nun endlich das Programm. **Vor dem Eintippen ist IMP INT: IMP FPT X-Y einzugeben!!**

Hochauflösende Druckerplots

```
10    CLEAR 32767:POKE #131,1
20    HZOLL!=8.0:VZOLL!=10.0:RATIO!=VZOLL!/HZOLL!
      (Maße des bedruckbaren Papierbereichs)
30    HRES=1920:VRES=1440
      (Bedruckbare Grafikpunkte horizontal und vertikal)
35    V_HM_HOEHE=16:V_NM_HOEHE=8:P_VERHAELT!=(RATIO!*HRES)/VRES:
       H_HM_HOEHE=P_VERHAELT!*V_HM_HOEHE:H_NM_HOEHE=P_VERHAELT!*
       V_NM_HOEHE
      (Höhen der Achsenhaupt- und Nebenmarkierungsstriche in Punkten. P_VERHAELT! ist das Verhältnis der
      waagerechten und senkrechten Punktmaße)
40    PREF=4:REM Anzahl der Bytes in der Druckerkontrollsequenz um den Drucker in Grafikmodus zu
      versetzen
50    DIM BUFF(HRES/8.0,1.0):BUFF=PREF+VARPTR(BUFF(0.0,0.0))
      (Puffer für Bitmusterdaten für eine Zeile hochauflösende Druckergrafik, einschließlich Kontroll-
      sequenz zum Einschalten des Grafikmodus)
60    BUFF(0.0,0.0)=#1B5A8007:REM Kontrollsequenz für 1920 Punkte höchstauflösende
      Grafik, Epson FX-80
      Für andere Drucker abändern!
70    DIM CLR(3.0):CLR=VARPTR(CLR(0.0))
      (Folgende Zeilen speichern in CLR folgendes Maschinenprogramm, das den Puffer BUFF leer macht):
      LXI D,HRES; LXI H,BUFF;CLRLUP MVI M,0; DCX D; INX H; MOV A,D; ORA E; RZ; JMP CLRLUP
80    CLR(0.0)=#11800721:CLR(1.0)=#3600:CLR(2.0)=#1B237AB3:
       CLR(3.0)=#C8C30000
90    POKE CLR+1,HRES IAND 255:POKE CLR+2,HRES/256
100   POKE CLR+4,BUFF IAND 255:POKE CLR+5,BUFF SHR 8
110   POKE CLR+14,(CLR+6) IAND 255:POKE CLR+15,(CLR+6)/256
120   DIM FLL(8.0):FLL=VARPTR(FLL(0.0))
      (Folgende Zeilen speichern in FLL folgendes Maschinenprogramm, das einen Bereich im Puffer BUFF mit
      einem Bitmuster füllt. Der Aufruf geschieht mit CALLM FLL: DOT VON,ZU BYTE. Das Bitmuster BYTE wird
      mit allen Stellen im Puffer zwischen VON (linke Grenze) und ZU (rechte Grenze) geodert; schon vor-
      handene Daten werden also nicht gelöscht! Nützlich, um horizontale gerade Striche zu ziehen.):
```

```
        INX B; CALL :E6F8 (berechne und hinterlasse in HL den 16-bit Wert eines Ausdrucks); XCHG; LXI H,BUFF;
        DAD D; PUSH H; CALL :E6F8; INX H; CALL :DE1A (HL-DE→HL); POP D; CALL :E71D (berechne und hinterlasse
        in A den 8-bit Wert eines Ausdrucks); PUSH B; MOV B,A;FLLLUP LDAX D; ORA B; STAX D; INX D; DCX H;
        MOV A,H; ORA L; JNZ FLLLUP; POP B; RET
130     FLL(0.0)=#3CDF8E6:FLL(1.0)=#EB210000:FLL(2.0)=#19E5CDF8
140     FLL(3.0)=#E623CD1A:FLL(4.0)=#DED1CD1D
150     FLL(5.0)=#E7C5471A:FLL(6.0)=#B012132B:FLL(7.0)=#7CB5C200:
          FLL(8.0)=#C1C900
160     POKE FLL+6,BUFF IAND 255:POKE FLL+7,BUFF SHR 8
170     POKE FLL+31,(FLL+23) IAND 255:POKE FLL+32,(FLL+23)/256
180     DIM MPRT(8.0):MPRT=VARPTR(MPRT(0.0))
        (Folgende Zeilen speichern in MPRT folgendes Maschinenprogramm, das den Puffer BUFF ausdruckt, d.h.
        an den Drucker schickt.  Ein eigenes Programm ist dafür nötig, weil das normale Basic Druckprogramm
        nach Grafikdaten mit dem Code #D des Wagenrücklaufs ein Byte #A (Zeilenvorschub) einfügen würde!):
    LXI D,HRES+PREF; LXI H,BUFF-PREF;MPLP1 LDA :FD00 (warte bis Drucker empfangsbereit); ANI 8;
        JZ MPLP1;MPLP2 LDA :FFF3 (warte bis serieller Ausgabepuffer leer); ANI :10; JZ MPLP2; MOV A,M;
        STA :FFF6 (serieller Ausgabepuffer); INX H; DCX D; MOV A,D; ORA E; RZ; JMP MPLP1
190     MPRT(0.0)=#11840721:MPRT(1.0)=#3A00
200     POKE MPRT+1,(HRES+PREF) IAND 255:POKE MPRT+2,
          (HRES+PREF)/256
210     POKE MPRT+4,(BUFF-PREF) IAND 255:POKE MPRT+5,
          (BUFF-PREF) SHR 8
220     MPRT(2.0)=#FDE608CA:MPRT(3.0)=#3AF3:POKE MPRT+12,
          (MPRT+6) IAND 255:POKE MPRT+13,(MPRT+6)/256
230     MPRT(4.0)=#FFE610CA:MPRT(5.0)=#7E32:POKE MPRT+20,
          (MPRT+14) IAND 255:POKE MPRT+21,(MPRT+14)/256
240     MPRT(6.0)=#F6FF231B:MPRT(7.0)=#7AB3C8C3:POKE MPRT+32,
          (MPRT+6) IAND 255:POKE MPRT+33,(MPRT+6)/256

1000    PDIM=VRES/16-1:REM Anzahl von Grafikdoppelzeilen (2x8 Bildpunkte hoch) auf dem Papier
1100    DIM PUNKTE!(PDIM,15.0),OBERGRENZE(PDIM,15.0),
          UNTERGRENZE(PDIM,15.0)
        PUNKTE! enthält die tatsächlichen Funktionswerte, UNTER- und OBERGRENZE hingegen geben an, von wo
        bis wo (in Bildpunkten gerechnet) die Graphenkurve sich in jeder Punktspalte des Graphen erstreckt.
1200    INPUT "ENTLANG X-ACHSE SKALA VON";XNIEDRIG:INPUT " BIS";
          XHOCH:PRINT
1300    INPUT "ENTLANG Y-ACHSE SKALA VON";YNIEDRIG:INPUT " BIS";
          YHOCH:PRINT
        Diese Angaben bestimmen, welcher Ausschnitt (in x- und y-Richtung) des Graphen zu zeichnen ist;
        dieser Ausschnitt wird dann in papierfüllendem Maßstab gezeichnet.  Für eine Achse (x- oder y-)
        können ein oder beide Endpunkte vom Programm bestimmt werden--dazu gebe man den Wert 999 ein.  Der
        tatsächlich verwendete Wert wird aus dem Wert des anderen Endpunktes so bestimmt, daß diese Achse
        im gleichen Längenmaßstab gezeichnet wird, wie die Andere; sind beide Endpunkte unbestimmt, wird ein
        entsprechend skalierter symmetrischer Ausschitt um die Null genommen.  Diese Berechnungen, sowie die
        Skalierung der Bildpunkte entlang den Achsen, werden in den folgenden Zeilen vorgenommen:
1400    IF XNIEDRIG=999.0 THEN 2000:IF XHOCH=999.0 THEN 2000:IF
          XHOCH<=XNIEDRIG THEN 1200:XDIFF=XHOCH-XNIEDRIG
1500    XSCHRITT=XDIFF/VRES:XPROZOLL=XDIFF/VZOLL!:YACHSE=-1.0:IF
          XNIEDRIG<=0 THEN YACHSE=-XNIEDRIG/XSCHRITT
        YACHSE gibt die Bildpunktlage der y-Achse an; wenn YACHSE < 0 ist die y-Achse nicht sichtbar.
1600    IF XHOCH=0.0 THEN YACHSE=VRES-1
        Wenn die Achse am rechten Bildrand ist, soll sie auf jeden Fall sichtbar bleiben.
1650    IF YACHSE>=VRES THEN YACHSE=-1.0
1700    YDIFF=HZOLL!*XPROZOLL:IF YHOCH=999.0 THEN 1800:IF YNIEDRIG
          <>999.0 THEN 2100:YNIEDRIG=YHOCH-YDIFF:GOTO 2200
1800    IF YNIEDRIG=999.0 THEN YNIEDRIG=-YDIFF/2.0
1900    YHOCH=YNIEDRIG+YDIFF:GOTO 2200
2000    IF YNIEDRIG=999.0 THEN 1200:IF YHOCH=999.0 THEN 1200:IF
          YNIEDRIG>=YHOCH THEN 1200
2100    YDIFF=YHOCH-YNIEDRIG
2200    YSCHRITT=YDIFF/HRES:XACHSE=-1.0:IF YNIEDRIG<=0 THEN XACHSE=
          -YNIEDRIG/YSCHRITT
        XACHSE gibt die Bildpunktlage der x-Achse an; wenn XACHSE < 0 ist die x-Achse nicht sichtbar.
```

```
2300  IF YHOCH=0.0 THEN XACHSE=HRES-1
```
Wenn die Achse am oberen Bildrand ist, soll sie auf jeden Fall sichtbar bleiben.
```
2400  IF XACHSE>=HRES THEN XACHSE=-1.0
2500  IF XNIEDRIG=999 THEN 2600:IF XHOCH<>999 THEN 3000:XDIFF=
       YDIFF*RATIO!:XHOCH=XNIEDRIG+XDIFF:GOTO 2800
2600  XDIFF=YDIFF*RATIO!:IF XHOCH=999.0 THEN XHOCH=XDIFF/2.0
2700  XNIEDRIG=XHOCH-XDIFF
2800  XSCHRITT=XDIFF/VRES:XPROZOLL=XDIFF/VZOLL!:YACHSE=-1.0:IF
       XNIEDRIG<=0 THEN YACHSE=-XNIEDRIG/XSCHRITT
```
YACHSE gibt die Bildpunktlage der y-Achse an; wenn YACHSE < 0 ist die y-Achse nicht sichtbar.
```
2900  IF XHOCH=0.0 THEN YACHSE=VRES-1
```
Wenn die Achse am rechten Bildrand ist, soll sie auf jeden Fall sichtbar bleiben.
```
3000  IF YACHSE>=VRES THEN YACHSE=-1.0
```
Folgende Zeilen fragen danach, welche Einteilung auf den Achsen markiert werden soll. Die Achsen werden in zwei Abstufungen markiert--eine Haupteinteilung in größere Intervalle, durch längere Striche markiert, und eine feinere Unterteilung dieser Hauptintervalle, durch kürzere Striche markiert. Die Achsen werden nur markiert, wenn sie selber auch sichtbar sind. Wenn keine Markierung gewünscht wird, gebe man "999" für die Hauptintervalle ein. Wenn nur eine einstufige Markierung gewünscht wird, gebe man "1" ein für die Anzahl der Feinunterteilungen pro Hauptintervall.
```
3005  XHAUPT=-1.0:IF XACHSE<0.0 THEN 3050:PRINT :PRINT
       "HAUPTMARKIERUNG DER X-ACHSE ALLE WIEVIELE EINHEITEN?":
       INPUT XHAUPT:PRINT
3010  IF XHAUPT=999.0 THEN XHAUPT=-1.0
```
negativer Wert von XHAUPT bedeutet **keine Markierung der x-Achse**
```
3020  IF XHAUPT<0.0 THEN 3050:INPUT "FEINEINTEILUNG DER X-ACHSE
       WIE OFT PRO HAUPTINTERVALL";NXFEIN:PRINT
3030  IF NXFEIN<=0.0 THEN XHAUPT=-1.0:GOTO 3050
3035  XFEIN=XHAUPT/NXFEIN:XF=XNIEDRIG/XFEIN:XF%=XF:XF=-FRAC(XF):
       IF XF<0.0 THEN XF%=XF%+1:XF=XF+1.0
3040  XFEINSCHRITT=XFEIN/XSCHRITT:XF=XF*XFEINSCHRITT:IF YACHSE>=0
       THEN XF=YACHSE+XF%*XFEINSCHRITT
```
XF ist die Lage (in Bildpunkten gerechnet) des nächsten Markierungsstriches auf der *x*-Achse.
(Man beachte: wenn die y-Achse vorhanden ist, wird XF von ihr ausgehend errechnet, um zu vermeiden, daß durch Rundungsfehler der Markierungsstrich für *x* = 0 direkt neben aber nicht auf der y-Achse gezeichnet wird, was tatsächlich manchmal sonst vorkommt. Ein anderer Weg, dieses Problem zu umgehen, wäre einfach, den Markierungsstrich für *x*=0 nicht zu zeichnen, was kaum auffallen würde.)
XFEINSCHRITT ist der Abstand in Bildpunkten zwischen Feinmarkierungen auf der *x*-Achse.
XF% zählt die Markierungen, und gibt somit an, ob die Nächste eine Haupt- oder Feinmarkierung ist.
```
3050  YHAUPT=-1.0:IF YACHSE<0.0 THEN 3100:PRINT :PRINT
       "HAUPTMARKIERUNG DER Y-ACHSE ALLE WIEVIELE EINHEITEN?":
       INPUT YHAUPT:PRINT
3055  IF YHAUPT=999.0 THEN YHAUPT=-1.0
3060  IF YHAUPT<0.0 THEN 3100: INPUT "FEINEINTEILUNG DER Y-ACHSE
       WIE OFT PRO HAUPTINTERVALL";NYFEIN:PRINT
3070  IF NYFEIN<=0.0 THEN YHAUPT=-1.0:GOTO 3100
3075  YFEIN=YHAUPT/NYFEIN:YF=YNIEDRIG/YFEIN:YF%=YF:YF=-FRAC(YF):
       IF YF<0.0 THEN YF%=YF%+1:YF=YF+1.0
3080  YFEINSCHRITT=YFEIN/YSCHRITT:YF=YF*YFEINSCHRITT:IF XACHSE>=0
       THEN YF=XACHSE+YF%*YFEINSCHRITT
```
YF ist die Lage (in Bildpunkten gerechnet) des ersten Markierungsstriches auf der y-Achse.
(Man beachte: wenn die *x*-Achse vorhanden ist, wird YF von ihr ausgehend errechnet, um zu vermeiden, daß durch Rundungsfehler der Markierungsstrich für y = 0 direkt neben aber nicht auf der *x*-Achse gezeichnet wird, was tatsächlich manchmal sonst vorkommt. Ein anderer Weg, dieses Problem zu umgehen, wäre einfach, den Markierungsstrich für y=0 nicht zu zeichnen, was kaum auffallen würde.)
YFEINSCHRITT ist der Abstand in Bildpunkten zwischen Feinmarkierungen auf der y-Achse.
YF% zählt die Markierungen, und gibt somit an, ob sie Haupt- oder Feinmarkierungen sind.

Hier Berechnung der Funktionswerte und der Daten des zu druckenden Graphen:
```
3100  FOR I=0 TO PDIM:FOR J=0 TO 15:ZAEHLER!=I*16.0+J
3200  X=XNIEDRIG+(ZAEHLER!+0.5)*XSCHRITT
```

Von hier bis Zeile 3300 steht die Berechnung der zu plottenden Funktion, wobei ein Wert pro Bildpunktspalte (da der Graph in Querformat gezeichnet wird, heißt das: pro Druckerpunktzeile) zu berechnen ist. Diese Werte müssen in Bildpunkteinheiten, gezählt vom linken Blattrand, umgewandelt werden und in diesen Einheiten in das Array PUNKTE! eingetragen werden. Die hier angegebene Funktion ist der hyperbolische Tangens; für andere Funktionen lösche man Zeile 3210 und ersetze man den Ausdruck (EPX!-EMX!)/(EPX!+EMX!) in Zeile 3300 durch einen neuen Ausdruck für f(X).

```
3210  EPX!=EXP(X):EMX!=1.0/EPX!
3300  PUNKTE!(I,J)=((EPX!-EMX!)/(EPX!+EMX!)-YNIEDRIG)/YSCHRITT:
       NEXT J:NEXT I
```

In den folgenden Zeilen wird aus den Werten in PUNKTE! ermittelt, in welchem Bereich der Graph jede Bildpunktzeile auf dem Papier schneidet, nach dem Prinzip: die Werte in PUNKTE! werden auf der Mittellinie der Bildpunktzeilen angenommen, und man erzeugt aus diesen berechneten Punkten den Graphen durch lineare Interpolation, d.h. indem man aufeinanderfolgende berechnete Punkte durch Geradenstücke verbindet. Da die berechneten Punkte selber auf der Mittellinie der Druckerpunktzeilen angenommen werden, wird von diesen verbindenden Geraden gerade eine Hälfte in einer Druckerpunktzeile sichtbar sein, und die andere Geradenhälfte erscheint in der benachbarten Punktzeile. Das Programm berechnet (als ganze Zahlen) die Endpunkte der in jeder Druckerpunktzeile sichtbaren Geradenhälften, wobei so gerundet wird, daß die Übergänge zwischen Geradenhälften ohne Lücken erscheinen und so, daß in jeder Geradenhälfte auf jeden Fall mindestens ein Punkt sichtbar bleibt. Die Ober- und Untergrenze des insgesamt in jeder Druckerpunktzeile entstehenden Strichs werden in den Arrays OBERGRENZE und UNTERGRENZE gespeichert; man beachte, daß dieser Gesamtstrich auch aus zwei Teilstrichen entsteht, nämlich aus jeweils einer Hälfte der Verbindungsgeraden von dem berechneten Punkt in dieser Bildpunktzeile zu den beiden benachbarten berechneten Punkten.

```
3400  ALT!=PUNKTE!(0.0,0.0):IALT=0.0:JALT=0.0:FOR I=0 TO PDIM:
       FOR J=0 TO 15
3500  NEU!=PUNKTE!(I,J):MITTELWERT=(ALT!+NEU!)/2.0:RGRENZE=NEU!+
       0.5:IF ALT!>NEU! THEN 4100
3600  BRUCH_LINKS=MITTELWERT:BRUCH_RECHTS=BRUCH_LINKS+1:IF
       RGRENZE=BRUCH_LINKS THEN BRUCH_RECHTS=BRUCH_LINKS
3700  IF LGRENZE>BRUCH_LINKS THEN BRUCH_LINKS=BRUCH_LINKS+1
3800  IF BRUCH_LINKS>OBERGRENZE(IALT,JALT) THEN OBERGRENZE(IALT,
       JALT)=BRUCH_LINKS
4000  OBERGRENZE(I,J)=RGRENZE:UNTERGRENZE(I,J)=BRUCH_RECHTS:
       GOTO 4600
4100  BRUCH_RECHTS=MITTELWERT:BRUCH_LINKS=BRUCH_RECHTS+1:IF
       LGRENZE=BRUCH_RECHTS THEN BRUCH_LINKS=BRUCH_RECHTS
4200  IF RGRENZE>BRUCH_RECHTS THEN BRUCH_RECHTS=BRUCH_RECHTS+1
4400  IF BRUCH_LINKS<UNTERGRENZE(IALT,JALT) THEN UNTERGRENZE(
       IALT,JALT)=BRUCH_LINKS
4500  OBERGRENZE(I,J)=BRUCH_RECHTS:UNTERGRENZE(I,J)=RGRENZE
4600  ALT!=NEU!:IALT=I:JALT=J:LGRENZE=RGRENZE:NEXT J:NEXT I
```

Jetzt wird gedruckt!

```
4700  POKE #131,0
4750  YACHSE%=INT(YACHSE)
4800  FOR I=0 TO PDIM:FOR K=0 TO 1
4900  CALLM CLR: REM Puffer löschen
5000  IF XACHSE>=0 THEN POKE BUFF+XACHSE,255: REM x-Achse setzen
5100  MASKE=#100:FOR J=K TO K+14 STEP 2:MASKE=MASKE SHR 1:
       ZAEHLER=16*I+J
```

In jeder Grafikzeile, die an den Drucker geschickt wird, erscheint immer nur jede zweite Bildpunktzeile--deshalb STEP 2. Die dazwischenliegenden Punktreihen werden mit einer neuen Grafikzeile gedruckt, nach einem Zeilenvorschub von einer halben Punkthöhe!
MASKE ist das Bitmuster für die gegenwärtige Bildpunktreihe im Grafikpuffer.

Nun wird geprüft, ob in dieser Punktreihe ein Markierungsstrich für die x-Achse vorkommt. Wenn beim höchsten zu plottenden x-Wert ein Markierungsstrich fällig ist, wird dieser auch noch gezeichnet in der letzten Punktreihe (der (VRES-1)-ten), obwohl durch die Art der Skalierung dieser x-Wert normalerweise in Punktreihe VRES vorkommt und somit gerade nicht mehr sichtbar wäre.

```
5210  IF XACHSE<0 THEN 5250:IF ZAEHLER=INT(XF) THEN 5215:IF
       ZAEHLER<>VRES-1.0 THEN 5250:IF VRES<>INT(XF) THEN 5250
5215  MA=XACHSE-H_NM_HOEHE:ME=XACHSE+H_NM_HOEHE:IF (XF% MOD
       NXFEIN)=0 THEN MA=XACHSE-H_HM_HOEHE:ME=XACHSE+H_HM_HOEHE
```

Hier wurden die Endpunkte des Markierungsstriches bestimmt. Er wird am Bildrand abgeschnitten wenn er über den Bildrand hinausgeht:

```
5220  IF MA<0 THEN MA=0
5230  IF ME>=HRES THEN ME=HRES-1
```

Nun zeichne die Markierung im Puffer und setze die Zeiger auf die nächste Markierung:

```
5240  CALLM FLL:DOT MA,ME MASKE:XF=XF+XFEINSCHRITT:XF%=XF%+1
```

Wenn diese Punktreihe der y-Achse entspricht, mache einen durchgehenden Strich, fertig:

```
5250  IF ZAEHLER=YACHSE% THEN STRICH_ANFANG=0:STRICH_ENDE=HRES-1:
       GOTO 5500
```

Sonst prüfe, ob diese Punktreihe so nahe bei der y-Achse liegt, daß sie die Markierungsstriche der y-Achse trifft, und wenn ja, setze die entsprechenden Punkte im Puffer durch POKEs (auch hier wird eine eventuelle Markierung für den höchsten y-Wert noch gezeichnet, obwohl sie normalerweise gerade nicht mehr sichtbar wäre):

```
5255  IF YACHSE%<0 THEN 5300:IF ZAEHLER<YACHSE%-V_HM_HOEHE THEN
       5300:IF ZAEHLER>YACHSE%+V_HM_HOEHE THEN 5300
5260  HAUPT=1:IF ZAEHLER>=YACHSE%-V_NM_HOEHE THEN IF ZAEHLER<=
       YACHSE%+V_NM_HOEHE THEN HAUPT=0
5270  MARK_ZAEHLER=YF%:MARK_Y!=YF
5280  IF MARK_Y!>HRES THEN 5300:IF HAUPT=1.0 THEN IF
       (MARK_ZAEHLER MOD NYFEIN)<>0.0 THEN 5290
5282  ADDR=BUFF+MARK_Y!:IF MARK_Y!=HRES THEN ADDR=BUFF+HRES-1
5286  POKE ADDR,PEEK(ADDR) IOR MASKE
5290  MARK_Y!=MARK_Y!+YFEINSCHRITT:MARK_ZAEHLER=MARK_ZAEHLER+1:
       GOTO 5280
```

Nun zeichne den eigentlichen Graphen in dieser Punktreihe (wenn er überhaupt sichtbar ist). Er wird am Bildrand abgeschnitten, wenn er über diesen hinausgeht. Punkte, die dem höchsten zu plottenden y-Wert (genauer gesagt, der ersten nicht mehr sichtbaren Punktspalte) entsprechen, werden in der letzten Punktspalte noch gezeichnet.

```
5300  STRICH_ANFANG=UNTERGRENZE(I,J):STRICH_ENDE=OBERGRENZE(I,J):
       IF STRICH_ANFANG<0.0 THEN STRICH_ANFANG=0:IF STRICH_ENDE<0
       THEN 5600
5400  IF STRICH_ENDE>=HRES THEN STRICH_ENDE=HRES-1:IF
       STRICH_ANFANG>HRES THEN 5600:IF STRICH_ANFANG=HRES THEN
       STRICH_ANFANG=HRES-1
5500  CALLM FLL:DOT STRICH_ANFANG,STRICH_ENDE MASKE
5600  NEXT J
```

Nun ist der Ausschnitt im Grafikpuffer fertig gezeichnet, also schicke ihn an den Drucker:

```
5700  CALLM MPRT
```

und mache einen Zeilenvorschub, abwechselnd 1/2 Punkthöhe und dann 7 1/2 Punkte (insgesamt eine 8-Punkt Grafikzeile hoch, doppelt dicht bedruckt!). Die hier angegebene PRINT Zeile ist für den Epson FX-80 und **muß für andere Drucker angepaßt werden!** Man beachte, daß der FX-80 eigentlich keine Zeilenvorschübe in halber Punkthöhe (1/144 Zoll) machen kann, sondern nur in Vielfachen von 1/3 Punkthöhe (1/216 Zoll)--hier werden deshalb in Wirklichkeit Zeilenvorschübe von abwechselnd 1/3 und 7 2/3 Punkthöhe gemacht (K ist abwechselnd 0 und 1), was trotzdem sehr gute Ergebnisse liefert. Man beachte auch: in der angegebenen PRINT Zeile wird zuerst der Zeilenabstand auf die gewünschte Größe gesetzt und dann ein Zeilenvorschub gemacht. Es ist wichtig zu bemerken, daß diese Befehlsfolge beim FX-80 gleichzeitig einen Wagenrücklauf bewirkt (im Gegensatz zum Befehl ESC J des FX-80, der nur Papiervorschub bewirkt). Wenn dieser Programmschritt für andere Drucker angepaßt wird, muß der Druckkopf auf jeden Fall an den Anfang einer neuen Druckzeile gebracht werden.

```
5800  PRINT CHR$(27);"3";CHR$(1+K*22);CHR$(10);
5900  NEXT K:REM Nächste Hälfte der doppeltdicht bedruckten Grafikzeile.
6000  NEXT I:REM Nächste doppeltdichte Grafikzeile
```

Nun ist das Bild fertig. Setze den Drucker wieder auf normalen Zeilenabstand (1/6 Zoll), mache einen Seitenvorschub, schalte die Druckausgabe ab, und END.
(Diese PRINT Zeile ist wieder für den FX-80 gedacht, und **muß für andere Drucker angepaßt werden!**)

```
6100  PRINT CHR$(27);"2";CHR$(12);:POKE #131,1:END: REM Grafik aus.
```

Bochum, den 16.1.1986

Literatur:

Mark Bridger, Mark Goresky: High-Resolution Printer Graphics, *BYTE*, November 1985, 219-232.

GP 26,11

GP 26,10

AUTONUMBER

Dieses Programm erzeugt bei Bedarf automatisch fortlaufende Zeilennummern. Die Anfangszeilennummer und die Schrittweite können bei Bedarf durch POKE festgelegt werden.Vor der Anwendung sollte der HEAP-Bereich höher gelegt werden um eine versehentliche Löschung des Programms zu vermeiden. Außer der Auto-Funktion ermöglicht das Programm die Eingabe über eine externe (serielle) Tastatur. Schließen Sie hierzu die Tastatur an die serielle Schnittstelle an. Die Break-Funktion gilt nur für die Eingabe. Ein Programm läßt sich daher nur bei einer Eingabe-Funktion oder durch die normale Break-Taste des DAI unterbrechen!

Da die Floppy-Station und die MDCR den Sprungvektor 'DINC' (#2E0) verwenden, ist diese Funktion noch anzupaßen.

Zur Benutzung: Poken Sie in die Speicherstellen 'LINENR' die erste zu verwendende Zeilennummer und in 'STEP' die gewünschte Schrittweite (unteres Byte= Wert modulo 256,obere Adresse=Wert/256 z.B. 'POKE #2F0,10:POKE#2F1,0' für Wert=10).

Anschließend poken Sie in die Speicherstelle 'INSW' (#296) einen der folgenden Werte:

0 - Eingabe über interne Tastatur, Auto aus
1 - Eingabe über externe Tastatur, Auto aus
4 - Auto ein, Eingabe über interne Tastatur
5 - Auto ein, Eingabe über externe Tastatur

Ist Auto eingeschaltet, so wird bei betätigen der Leertaste in der ersten Eingabeposition die Zeilennummer ausgegeben. Wird statt dessen eine andere Taste betätigt, so wird die Zeilennummer unterdrückt.
Dadurch braucht die Zeilennummer nicht nochmals gelöscht werden, wenn man eine Zeile einfügen will.

Np1

```
002                     START   EQU     :02F0       EVTL. ANPASSEN
003                     RUNF    EQU     :0118       0=EINGABE, 1=PROGRAMM
004                     DINC    EQU     :02E0       SPRUNG ZUR EINGABE
005                     INSW    EQU     :0296       EINGABE-SCHALTER
006                     PRTNR   EQU     :EFB4       GIEBE (HL) ALS DEZ. ZAHL A'S
007                     GETC    EQU     :D4E2       HOLE ZEICHEN VON TASTATUR
008                     GETRS   EQU     :DDB4       HOLE ZEICHEN VON RS-232
009                     GETBRS  EQU     :DDC0       HOLE BREAK-STATUS VON RS-232
010
011                     AUTOM   EQU     :04         MASKE FUER AUTO (BIT 2)
012                     INPUTM  EQU     :01         MASKE FUER TASTATUR/RS-232
013                     MDCRM   EQU     :00         MASKE FUER MDCR/FLOPPY
014                                                 FUER MEMOCOM-MDCR = :02
015                     MDCR    EQU     :0000       SPRUNGADR. FUER MDCR/FLOPPY
016                                                 WERT ENTSPRICHT URSPRUENG-
017                                                 LICHEM WERT DES DINC-SPRUNGS
018                                                 BEI VERWENDUNG VON FLOPPY
019                                                 ODER MDCR EVTL. ANPASSEN
020
021                             ORG     DINC
022 02E0 C3F402                 JMP     AUTO
023
024                             ORG     START
025 02F0 0A00           LINENR  DBL     10          SPEICHER FUER ZEILENNR.
026 02F2 0A00           STEP    DBL     10          SPEICHER FUER SCHRITTWEIT
027
028 02F4 E600           AUTO    ANI     MDCRM
029 02F6 C20000                 JNZ     MDCR
030 02F9 3A9602                 LDA     INSW
031 02FC E604                   ANI     AUTOM
032 02FE CA3503                 JZ      GETKEY
033 0301 3A1801                 LDA     RUNF        LAEUFT PROGRAMM ?
034 0304 B7                     ORA     A
035 0305 C23503                 JNZ     GETKEY
036 0308 E5                     PUSH    H
037 0309 EF                     RST     5           LADE CURSORPOS. IN HL
038 030A 0C                     DATA    :0C         L=X-POSITION
039 030B 2D                     DCR     L
040 030C C23403                 JNZ     DONE2
041 030F CD3503                 CALL    GETKEY
042 0312 DA3203                 JC      DONE
043 0315 FE20                   CPI     ' '         TASTE = SPACE ?
044 0317 C23103                 JNZ     DONE1
045 031A C5                     PUSH    B
046 031B D5                     PUSH    D
047 031C 2AF002                 LHLD    LINENR
048 031F CDB4EF                 CALL    PRTNR       GEBE ZEILENNR. AUS
049 0322 2AF202                 LHLD    STEP        BERECHNE NAECHSTE ZEILENNR.
050 0325 EB                     XCHG
051 0326 2AF002                 LHLD    LINENR
052 0329 19                     DAD     D
053 032A 22F002                 SHLD    LINENR
054 032D 3E20                   MVI     A,' '
055 032F D1                     POP     D
056 0330 C1                     POP     B
057 0331 B7             DONE1   ORA     A
058 0332 E1             DONE    POP     H
059 0333 C9                     RET
060 0334 E1             DONE2   POP     H
061 0335 3A9602         GETKEY  LDA     INSW
062 0338 E601                   ANI     INPUTM
063 033A CAE2D4                 JZ      GETC        EINGABE UEBER INT. TASTAT
064 033D CDC0DD                 CALL    GETBRS      BREAK VON EXT. TASTATUR ?
065 0340 D8                     RC
066 0341 C3B4DD                 JMP     GETRS       HOLE CHR. VON EXT. TASTATUR
067
068 0344                        END
```

```
59000 REM BASIC Interface Formatdruck und -Listen
59010 REM Legt das Maschinenprogramm unterhalb des
59020 REM Heap ab und belegt die Parameter.
59030 REM (C) Bernd Preusing    22.3.82
59040 REM
59100 CLEAR 1000
59110 POKE #29B,#7C:POKE #29C,#4:REM START HEAP
59120 CLEAR #100
59130 POKE #131,1
59140 POKE #2DD,#C3:POKE #2DE,#30:POKE #2DF,#3
59200 FOR ADR%=#330 TO #47B:READ BYTE%:POKE ADR%,BYTE%:NEXT
59210 FOR ADR%=#2EC TO #32F:POKE ADR%,0:NEXT
59260 PRINT CHR$(12):PRINT :PRINT
59265 INPUT "Locherrand";LR%:PRINT
59267 POKE #2EE,LR%
59270 INPUT "Zeichen pro Zeile incl. Rand";ZZ%:PRINT
59275 IF ZZ%>0 THEN POKE #2EC,ZZ%
59280 INPUT "Zeilen pro Seite";ZS%:PRINT
59285 IF ZS%>0 THEN POKE #2ED,ZS%
59290 INPUT "Titelzeile u./o. Seitennummern <J/N>";T$:PRINT

59300 IF T$="N" THEN 59500
59310 IF T$<>"J" THEN 59290
59320 POKE #2F2,#A:SFLAG%=0
59330 PRINT "Bitte geben Sie den Titel ein,"
59335 PRINT "fuer die Seitennummer druecken Sie '&' !"
59340 INPUT TITEL$:PRINT
59345 IF LEN(TITEL$)>55 THEN 59330
59350 ADR%=#2F8
59355 FOR I%=0 TO LEN(TITEL$)-1
59360 BYTE%=ASC(MID$(TITEL$,I%,1))
59365 IF BYTE%=ASC("&") THEN SFLAG%=1
59370 POKE ADR%,BYTE%:ADR%=ADR%+1
59380 NEXT I%
59390 POKE ADR%,0
59400 INPUT "Leerzeilen zwischen Titel u. Text";LZTT%:PRINT

59402 POKE #2EF,LZTT%
59405 IF SFLAG%=0 THEN 59450
59410 INPUT "Erste Seitennummer";SNR%:PRINT
59415 IF SNR%<1 OR SNR%>9999 THEN 59410
59420 SNR$=MID$(STR$(SNR%),1,LEN(STR$(SNR%))-3)
59430 FOR ADR%=#2F3 TO #2F6:POKE ADR%,ASC("0"):NEXT
59440 FOR I%=LEN(SNR$) TO 1 STEP (-1):ADR%=ADR%-1:POKE ADR%,
      ASC(MID$(SNR$,I%-1,1)):NEXT
59450 INPUT "Titeltext rechts, links, mittig  <L/R/M>";P$:PR
      INT
59460 IF P$="L" THEN POKE #2F7,0:GOTO 59500
59470 IF P$="R" THEN POKE #2F7,ZZ%-LR%-LEN(TITEL$)-2:GOTO 59
      500
59480 IF P$="M" THEN POKE #2F7,(ZZ%-LR%-LEN(TITEL$))/2:GOTO
      59500
59490 GOTO 59450
59500 PRINT :PRINT "Aufruf der Formatierung durch POKE #131,
      4:PRINT.. oder :LIST":PRINT :PRINT
59999 REM #330-#47B
60000 DATA #F5,#3A,#31,#01,#FE,#04,#DA,#44
60010 DATA #03,#CA,#4C,#03,#FE,#06,#DA,#48
60020 DATA #03,#F1,#C9,#00,#F1,#C3,#94,#DD
60030 DATA #F1,#C3,#94,#DD,#F1,#EF,#03,#FE
```

```
60040 DATA #0D,#CA,#88,#03,#FE,#0C,#CA,#AC
60050 DATA #03,#E5,#F5,#3A,#F2,#02,#1F,#D4
60060 DATA #CC,#03,#1F,#0C,#FD,#03,#1F,#0C
60070 DATA #ED,#03,#F1,#CD,#49,#03,#F5,#21
60080 DATA #F0,#02,#34,#3A,#EC,#02,#BE,#CC
60090 DATA #7D,#03,#F1,#E1,#C9,#3E,#0D,#CD
60100 DATA #8B,#03,#3A,#F2,#02,#F6,#05,#32
60110 DATA #F2,#02,#C9,#F5,#E5,#CD,#49,#03
60120 DATA #AF,#32,#F0,#02,#3A,#F2,#02,#E6
60130 DATA #FA,#32,#F2,#02,#21,#F1,#02,#34
60140 DATA #3A,#ED,#02,#BE,#CC,#AA,#03,#E1
60150 DATA #F1,#C9,#3E,#0C,#F5,#CD,#49,#03
60160 DATA #AF,#32,#F0,#02,#32,#F1,#02,#3A
60170 DATA #F2,#02,#E6,#08,#3A,#F2,#02,#C4
60180 DATA #C9,#03,#E6,#FA,#32,#F2,#02,#F1
60190 DATA #C9,#F6,#02,#C9,#F5,#3A,#EE,#02
60200 DATA #32,#F0,#02,#B7,#CA,#E1,#03,#6F
60210 DATA #3E,#20,#CD,#49,#03,#2D,#C2,#DA
60220 DATA #03,#3A,#F2,#02,#F6,#01,#E6,#FB
60230 DATA #32,#F2,#02,#F1,#C9,#F5,#CD,#CC
60240 DATA #03,#2E,#06,#3A,#F0,#02,#85,#32
60250 DATA #F0,#02,#C3,#D8,#03,#17,#DC,#CC
60260 DATA #03,#F5,#3A,#F2,#02,#E6,#FD,#32
60270 DATA #F2,#02,#CD,#13,#04,#F1,#D4,#CC
60280 DATA #03,#1F,#C9,#E5,#D5,#21,#F7,#02
60290 DATA #7E,#B7,#CA,#27,#04,#57,#3E,#20
60300 DATA #CD,#49,#03,#15,#C2,#20,#04,#23
60310 DATA #7E,#FE,#26,#CA,#49,#04,#B7,#CA
60320 DATA #38,#04,#CD,#49,#03,#C3,#27,#04
60330 DATA #3A,#EF,#02,#3C,#57,#3E,#0D,#CD
60340 DATA #8B,#03,#15,#C2,#3F,#04,#D1,#E1
60350 DATA #C9,#E5,#21,#F3,#02,#16,#04,#1E
60360 DATA #30,#7E,#23,#BB,#C2,#5C,#04,#15
60370 DATA #C2,#51,#04,#14,#CD,#49,#03,#7E
60380 DATA #23,#15,#C2,#5C,#04,#CD,#6C,#04
60390 DATA #E1,#C3,#27,#04,#16,#04,#2B,#2B
60400 DATA #34,#7E,#FE,#3A,#D8,#36,#30,#15
60410 DATA #C2,#6F,#04,#C9
```

```
59000 REM BASIC Interface Formatdruck und -Listen
59010 REM Legt das Maschinenprogramm unterhalb des
59020 REM Heap ab und belegt die Parameter.
59030 REM (C) Bernd Preusing   22.3.82
59040 REM .   relativiert    19.6.82
59050 REM
59060 REM PLATZ FUER DAS ML-PROGRAMM MACHEN:
59070 REM
59100 CLEAR 1000
59110 OLDHEAP=PEEK(#29B)+256*PEEK(#29C)
59120 NEUHEAP=OLDHEAP+#190
59130 POKE #29B,NEUHEAP IAND #FF
59140 POKE #29C,NEUHEAP SHR 8
59150 POKE #29D,(1000-#190) IAND #FF
59160 POKE #29E,(1000-#190) SHR 8
59170 POKE NEUHEAP,((1000-#190-4) SHR 8) IOR #80
59180 POKE NEUHEAP+1,(1000-#190-4) IAND #FF
59190 REM
59200 REM PROGRAMM EINLESEN
59210 REM
59220 POKE #131,1:REM NUR AUF SCREEN
59230 START=OLDHEAP+#44
59240 POKE #2DD,#C3:POKE #2DE,START IAND #FF:POKE #2DF,START SHR 8

59250 FOR ADR=START TO NEUHEAP-1:READ BYTE:POKE ADR,BYTE:NEXT
59260 FOR ADR=OLDHEAP TO START-1:POKE ADR,0:NEXT
59270 REM
59280 REM ABSOLUTE ADRESSEN BERECHNEN
59290 REM
59300 READ NREL
59310 FOR I=1 TO NREL
59320 READ POSREL
59330 MOM=POSREL+START
59340 ALTWERT=PEEK(MOM)+256*PEEK(MOM+1)
59350 POKE MOM,(ALTWERT+OLDHEAP) IAND #FF
59360 POKE MOM+1,(ALTWERT+OLDHEAP) SHR 8
59370 NEXT
59380 REM
59390 REM PARAMETER ERFRAGEN
59400 REM
59410 PRINT CHR$(12):PRINT :PRINT
59420 INPUT "Locherrand";LR:PRINT
59430 POKE OLDHEAP+2,LR
59440 REM
59450 INPUT "Zeichen pro Zeile incl. Rand";ZZ:PRINT
59460 IF ZZ>0 THEN POKE OLDHEAP,ZZ
59470 REM
59480 INPUT "Zeilen pro Seite";ZS:PRINT
59490 IF ZS>0 THEN POKE OLDHEAP+1,ZS
59500 REM
59510 INPUT "Titelzeile u./o. Seitennummern <J/N>";T$:PRINT
59520 IF T$="N" GOTO 59880
59530 IF T$<>"J" GOTO 59510
59540 REM
59550 POKE OLDHEAP+6,#A
59560 SFLAG=0
59570 PRINT "Bitte geben Sie den Titel ein,"
59580 PRINT "fuer die Seitennummer druecken Sie '&' !"
59590 REM FUER EIN ANDERES ZEICHEN ALS '&' IN ZEILE
59600 REM 60310 DAS DRITTE DATA AENDERN.
59610 INPUT TITEL$:PRINT
59620 IF LEN(TITEL$)>55 GOTO 59570
59630 ADR=OLDHEAP+12
```

```
59640 FOR I=0 TO LEN(TITEL$)-1
59650 BYTE=ASC(MID$(TITEL$,I,1))
59660 IF BYTE=ASC("&") THEN SFLAG=1
59670 POKE ADR,BYTE
59680 ADR=ADR+1
59690 NEXT I
59700 POKE ADR,0:REM =TEXTENDE
59710 REM
59720 INPUT "Leerzeilen zwischen Titel u. Text";LZTT:PRINT
59730 POKE OLDHEAP+3,LZTT
59740 IF SFLAG=0 GOTO 59820
59750 REM
59760 INPUT "Erste Seitennummer";SNR:PRINT
59770 IF SNR<1 OR SNR>9999 GOTO 59760
59780 SNR$=MID$(STR$(SNR),1,LEN(STR$(SNR))-3)
59790 FOR ADR=OLDHEAP+7 TO OLDHEAP+10:POKE ADR,ASC("0"):NEXT
59800 FOR I=LEN(SNR$) TO 1 STEP (-1):ADR=ADR-1:POKE ADR,ASC(MID$(S
      NR$,I-1,1)):NEXT
59810 REM
59820 INPUT "Titeltext rechts, links, mittig  <L/R/M>";P$:PRINT
59830 IF P$="L" THEN POKE OLDHEAP+11,0:GOTO 59880
59840 IF P$="R" THEN POKE OLDHEAP+11,ZZ-LR-LEN(TITEL$)-2:GOTO 5988
      0
59850 IF P$="M" THEN POKE OLDHEAP+11,(ZZ-LR-LEN(TITEL$))/2:GOTO 59
      880
59860 GOTO 59820
59870 REM
59880 PRINT :PRINT "Aufruf der Formatierung durch POKE #131,4:PRIN
      T.. oder :LIST":PRINT :PRINT
59890 REM
59999 REM #0044-#018F     (= ORG :0)
60000 DATA #F5,#3A,#31,#01,#FE,#04,#DA,#58
60010 DATA #00,#CA,#60,#00,#FE,#06,#DA,#5C
60020 DATA #00,#F1,#C9,#00,#F1,#C3,#CF,#0A
60030 DATA #F1,#C3,#94,#DD,#F1,#EF,#03,#FE
60040 DATA #0D,#CA,#9F,#00,#FE,#0C,#CA,#C0
60050 DATA #00,#E5,#F5,#3A,#06,#00,#1F,#D4
60060 DATA #E0,#00,#1F,#DC,#11,#01,#1F,#DC
60070 DATA #01,#01,#F1,#CD,#5D,#00,#F5,#21
60080 DATA #04,#00,#34,#3A,#00,#00,#BE,#CC
60090 DATA #91,#00,#F1,#E1,#C9,#3E,#0D,#CD
60100 DATA #9F,#00,#3A,#06,#00,#F6,#05,#32
60110 DATA #06,#00,#C9,#F5,#E5,#CD,#5D,#00
60120 DATA #AF,#32,#04,#00,#3A,#06,#00,#E6
60130 DATA #FA,#32,#06,#00,#21,#05,#00,#34
60140 DATA #3A,#01,#00,#BE,#CC,#BE,#00,#E1
60150 DATA #F1,#C9,#3E,#0C,#F5,#CD,#5D,#00
60160 DATA #AF,#32,#04,#00,#32,#05,#00,#3A
60170 DATA #06,#00,#E6,#08,#3A,#06,#00,#C4
60180 DATA #DD,#00,#E6,#FA,#32,#06,#00,#F1
60190 DATA #C9,#F6,#02,#C9,#F5,#3A,#02,#00
60200 DATA #32,#04,#00,#B7,#CA,#F5,#00,#6F
60210 DATA #3E,#20,#CD,#5D,#00,#2D,#C2,#EE
60220 DATA #00,#3A,#06,#00,#F6,#01,#E6,#FB
60230 DATA #32,#06,#00,#F1,#C9,#F5,#CD,#E0
60240 DATA #00,#2E,#06,#3A,#04,#00,#85,#32
60250 DATA #04,#00,#C3,#EC,#00,#17,#DC,#E0
60260 DATA #00,#F5,#3A,#06,#00,#E6,#FD,#32
60270 DATA #06,#00,#CD,#27,#01,#F1,#D4,#E0
60280 DATA #00,#1F,#C9,#E5,#D5,#21,#0B,#00
60290 DATA #7E,#B7,#CA,#3B,#01,#57,#3E,#20
60300 DATA #CD,#5D,#00,#15,#C2,#34,#01,#23
60310 DATA #7E,#FE,#26,#CA,#5D,#01,#B7,#CA
```

```
60320 DATA #4C,#01,#CD,#5D,#00,#C3,#3B,#01
60330 DATA #3A,#03,#00,#3C,#57,#3E,#0D,#CD
60340 DATA #9F,#00,#15,#C2,#53,#01,#D1,#E1
60350 DATA #C9,#E5,#21,#07,#00,#16,#04,#1E
60360 DATA #30,#7E,#23,#BB,#C2,#70,#01,#15
60370 DATA #C2,#65,#01,#14,#CD,#5D,#00,#7E
60380 DATA #23,#15,#C2,#70,#01,#CD,#80,#01
60390 DATA #E1,#C3,#3B,#01,#16,#04,#2B,#2B
60400 DATA #34,#7E,#FE,#3A,#D8,#36,#30,#15
60410 DATA #C2,#83,#01,#C9
60500 REM
60510 REM POSITIONEN, AN DENEN RELATIVE ADR. STEHEN
60520 DATA 65
60530 DATA #07,#0A,#0F,#22,#27,#2C,#30,#34
60540 DATA #38,#3C,#40,#44,#48,#50,#53,#58
60550 DATA #5E,#62,#65,#6A,#6D,#71,#75,#7E
60560 DATA #82,#85,#88,#8D,#90,#95,#9E,#A1
60570 DATA #A5,#AB,#AF,#B2,#B9,#BF,#C4,#C8
60580 DATA #CB,#CF,#D3,#D8,#DB,#DF,#E6,#EB
60590 DATA #F1,#F5,#FC,#100,#103,#106,#109,#110
60600 DATA #114,#11B,#125,#129,#12D,#133,#136,#13A
60610 DATA #149
```