Grafik mit der Hercules-Karte unter Forth

Claus Kühnel*

Während für Turbo-Pascal und Turbo-C die gängigen Videokarten unterstützt werden, stehen solche Treiber unter Forth nicht ohne weiteres zur Verfügung. Am Beispiel der Hercules-Karte wird hier die Ansteuerung von Forth aus beschrieben.

1. Grundsätzliches

Aufgrund ihres modularen Hardwarekonzepts werden IBM-PC/XT/AT und kompatible Rechner vielerorts zur Lösung von Aufgaben der Messtechnik, Prozesssteuerung und anderem eingesetzt. Eine umfangreiche Produktpalette von PC-Einschubkarten steht für ein breites Spektrum von Anwendungsfällen zur Verfügung. Die freien Slots des PC nehmen die entsprechenden Interface-Karten auf und sichern damit die Kopplung zur «Aussenwelt».

Neben komfortablen und damit kostenintensiven Softwarepaketen werden sich gerade in der Mess- und Steuerungstechnik weiterhin «massgeschneiderte» Softwarelösungen behaupten müssen. Aufgrund der Spezifika nahezu jeder Applikation in diesem Bereich sind der Vervielfältigung eines Softwareprodukts oft enge Grenzen gesetzt. Komfortable, interaktive Programmierumgebungen wie Forth sind deshalb zur Erstellung individueller Software an diesen Stellen favorisiert [1].

Während die Fa. Borland mit den BGI-Treibern (Borland Graphic Interface) für Turbo-Pascal und Turbo-C die gängigen Videokarten unterstützt, stehen solche Treiber unter Forth nicht ohne weiteres zur Verfügung.

Am Beispiel der Hercules-Karte wird die einfache Ansteuerung von Forth aus verdeutlicht. Verwendet wird die verbreitete Laxen & Perry-Implementierung F83. Die Übertragung auf andere Systeme wird dadurch nicht eingeschränkt.

Für messtechnische Applikationen ist die Hercules-Karte eine sehr kostengünstige Alternative und zudem in einer Reihe von PCs von vornherein vorhanden. Unter diesen Voraussetzungen erscheint die Wahl als gerechtfertigt.

2. Anforderungen der Messtechnik an die Grafik

Jedes Anwendungsgebiet stellt eigene Anforderungen an die Grafik eines Computers, die ihrerseits die Auswahl einer entsprechenden Videokarte bestimmen. Der Einsatz der Hercules-Karte war an dieser Stelle vorweggenommen, weshalb bestimmte Eigenschaften auch vorab festgelegt sind.

Die Auswahl der Hercules-Karte legt

- monochrome Darstellung und
- Auflösung 720 × 348 Bildpunkte

fest. Diese «bit-mapped» Version des Monochromadapters MDA benötigt einen Bildwiederholspeicher von weniger als 32 KByte, weshalb auch zwei Seiten des Bildwiderholspeichers zur Verfügung stehen.

Aus den Bedingungen der Messtechnik resultiert die Forderung nach grafischer Darstellung von Funktionsverläufen, xy-Plots und anderes. Die Ansteuerung eines einzelnen Bildpunkts muss zur Realisierung ausreichend aufgelöster Grafiken möglich sein.

Die Verbindung zweier beliebiger Bildpunkte durch eine Gerade sollte ausserdem in vertretbarer Zeit geschehen. Neben der Darstellung von Punkten und Linien kommt der Beschriftung einer Grafik bzw. der Dialoggestaltung im Grafikmodus eine grosse Bedeutung zu.

Will man seine grafische Darstellung «mit nach Hause nehmen», dann sollte eine Hardcopy-Routine nicht fehlen.

Bei der Archivierung des Datenmaterials kann man unterschiedliche Wege gehen. Zum einen werden die erhobenen Daten auf einem externen Speichermedium abgespeichert und stehen damit bei Bedarf sofort erneut zur Verfügung. Andererseits lassen sich erzeugte Grafiken als Pixel-Datei abspeichern. Da der erste Weg im allgemeinen ohnehin gegangen wird, muss die Applikation über die Notwendigkeit von Pixel-Dateien entscheiden.

3. Programmierung der Hercules-Karte

Die Modi der Hercules-Karte sind durch die Software einzustellen. Hierzu befinden sich auf der Hercules-Karte eine Reihe von Registern, die entsprechend den Erfordernissen zu belegen sind. Auf die Einzelheiten hierzu soll nicht eingegangen werden, da diese praktisch standardisiert sind und zum anderen aus dem Listing 1 deutlich werden.

Wichtig zur Adressierung des einzelnen Bildpunkts ist der Zusammenhang zwischen Grafikkoordinate und Adresse im Bildwiederholspeicher. Gelesen und beschrieben wird generell ein **komplettes Byte** im Bildwiederholspeicher. Zum Setzen eines Bildpunkts ist deshalb zuerst das betreffende Byte (acht Bildpunkte) im Bildwiederholspeicher zu lesen. Dieser Wert ist nun entsprechend der geforderten Bitposition zu verändern (bitweises OR) und anschliessend wieder in den Bildwiederholspeicher zurückzuschreiben.

Mit den folgenden Beziehungen lassen sich die notwendigen Grössen berechnen:

Byte-Offset = 8192 * (Y mod 4) + 90 * int (Y/4) + int(X/8)
Bit-Offset = 8 − (X mod 8)

Die xy-Koordinaten sind im Grafikmodus dabei in der folgenden Weise festgelegt:

```
(X,Y) = (0,0)                              (719,0)
        ------------------------------------------
        :                                        :
        :                                        :
        :                                        :
        :                                        :
        :                                        :
        :                                        :
        ------------------------------------------
        (0,347)                          (719,347)
```

* Dr. Claus Kühnel, Zschertnitzerstr. 52, DDR-8020 Dresden

Glossarium

!anstieg (x1 y1 x2 y2 -- x1 y1 x2 y2)
Berechnung und Abspeicherung des Anstiegs der Geraden.

(hardcopy) (--)
Hardcopy-Routine - Lesen des Bildwiederholspeichers und Ausgabe an den Drucker.

1/anstieg (--)
Vertauschen von Zähler und Nenner des Anstiegs.

@maske&byte (xpos ypos -- maske addr offset b)
Bereitstellung von Maskierungsbyte (maske), gelesenem Byte (b) und Adresse im Bildwiederholspeicher (addr, offset) für das zu schreibende Byte.

anstieg (-- addr)
32-Bit Variable, die Zähler und Nenner des Anstiegs der Geraden enthält.

beeps (u --)
Ausgabe mehrerer akustischer Signale ("Beeps").

bis<von? (x1 y1 x2 y2 -- x1 y1 x2 y2 f)
Flag gleich true, falls x1 kleiner als x2 ist.

bitoffset (xpos -- offset)
Bestimmung des darzustellenden Bildpunktes innerhalb eines Bytes.

byteoffset (xpos ypos -- offset)
Bestimmung des Offsets zur Anfangsadresse des Bildwiederholspeichers.

bitpos (-- addr)
Array von Maskierungsbytes.

cleargraph (--)
Löschen des Grafikbildschirms.

configsw (-- addr)
Adresse des Konfigurationsregisters der Hercules-Karte.

datareg (-- addr)
Adresse des Datenregisters der Hercules-Karte.

esc (-- 16b)
Konstante, die den Wert für ESC (1Bh) übergibt.

f (--)
Seitenvorschub.

grparam (-- addr)
Parameterfeld für Grafikmodus.

grtest (--)
Testroutine - jeder vierte Bildpunkt des Grafikbildschirms wird umgeschaltet.

hardcopy (--)
Komplette Hardcopyroutine einschließlich Bedienerführung, Vektorisierung der Ausgabe und Initialisierung des Druckers.

indexreg (-- addr)
Adresse des Indexregisters der Hercules-Karte.

initgraph (--)
Umschalten in den Grafikmodus der Hercules-Karte.

init_gra_printer (--)
Einstellung von linkem Rand und erforderlichem Zeilenvorschub für Hardcopy-Ausgabe.

keine_senkrechte (x1 y1 x2 y2 --)
Darstellung einer Linie mit beliebigem, aber endlichem Anstieg.

l! (w segment offset --)
Schreiben eines 16-Bit-Eintrages in den segmentierten Speicher.

l@ (segment offset -- w)
Lesen eines 16-Bit-Eintrages aus dem segmentierten Speicher.

lc! (b segment offset --)
Schreiben eines Bytes in den segmentierten Speicher.

lc@ (segment offset -- b)
Lesen eines Bytes aus dem segmentierten Speicher.

line (x1 y1 x2 y2 --)
Darstellung einer Linie mit beliebigem Anstieg. Einhaltung des Gültigkeitsbereichs der Koordinaten wird **nicht** überprüft.

loop_ind (x1 y1 x2 y2 -- y1 x1 x2+1 x1)
Bereitstellung der Schleifenindizes für die Worte y(x)` und x(y).

modectrl (-- addr)
Adresse des Modusregisters der Hercules-Karte.

resdot (xpos ypos --)
Rücksetzen des Bildpunktes mit den Koordinaten (xpos,ypos).

resetgraph (--)
Umschalten in den Textmodus der Hercules-Karte.

seite_0 (-- addr)
Adresse der Seite 0 des Bildwiederholspeichers der Hercules-Karte.

seite_1 (-- addr)
Adresse der Seite 1 des Bildwiederholspeichers der Hercules-Karte.

senkrechte (x y1 x y2 --)
Darstellung einer senkrechten Linie.

setdot (xpos ypos --)
Setzen des Bildpunktes mit den Koordinaten (xpos,ypos).

setresbyte (xpos ypos f --)
Setzen (true-flag) bzw. Rücksetzen (false-flag) des die Position (xpos,ypos) beinhaltenden Bytes im Bildwiederholspeicher.

setresdot (xpos ypos f --)
Setzen (true-flag) bzw. Rücksetzen (false-flag) des Bildpunktes mit den Koordinaten (xpos,ypos).

switchdot (xpos ypos --)
Umschalten des Bildpunktes mit den Koordinaten (xpos,ypos).

txtparam (-- addr)
Parameterfeld für Textmodus.

txttable (-- addr offset)
Konstante, die die Adresse des IBM-Zeichensatzes übergibt.

wrchar (xpos ypos char --)
Ausgabe des Zeichens (char) an die Position (xpos,ypos).

wrtxt (xpos ypos addr count --)
Ausgabe eines im TYPE-Format (addr, count) vorliegenden Strings an die Position (xpos,ypos).

x(y) (x1 y1 x2 y2 --)
Darstellung einer Geraden mit einem Anstieg /m/ > 1 durch deren Umkehrfunktion.

x1=x2? (x1 y1 x2 y2 -- x1 y1 x2 y2 f)
Flag gleich true, falls x1 = x2 (senkrechte Linie!).

x_y_tausch (x1 y1 x2 y2 -- y1 x1 y2 x2)
Vertauschen der x-y-Koordinaten.

y(x) (x1 y1 x2 y2 --)
Darstellung einer Geraden mit einem Anstieg /m/ <= 1.

Glossarium zur Grafikerweiterung der Hercules-Karte mit Forth.

4. Die Grafikerweiterung HERCULES.BLK

Die Ansteuerung der Hercules-Karte wird durch die nachladbare Komponente **HERCULES.BLK** sichergestellt. Die Erweiterung beansprucht 1881 Byte im Wörterbuch des Forth-Systems und ist damit recht kompakt. Das Listing 1 zeigt mit den Screens # 0 bis # 18 die Bestandteile der Komponente. Auf die komfortable Darstellung mit Hilfe des Shadow-Screen-Konzepts wurde aus Gründen des hohen Platzbedarfs verzichtet. Die erforderlichen Kommentare sind dem Glossarium zu entnehmen.

Der Zugriff auf den segmentierten Speicher geschieht durch die Worte in Screen # 2. Diese arbeiten in der gleichen Weise wie die bekannten Worte @ und !, nur dass die Adressangabe durch Angabe von Segment und Offset ersetzt ist. Die in Screen # 3 angegebenen Registeradressen bzw. Grafik- und Textmodeparameter sind praktisch standardisiert und im Handbuch der Hercules-Karte zu finden. Nach diesen Vorgaben erfolgt auch die Initialisierung von Grafik- **(initgraph)** und Textmodus **(resetgraph)** in den Screens # 4 und # 5. Damit das Löschen des Grafikbildschirms hinreichend schnell geschieht, wurde das Wort **cleargraph** in Screen # 6 als Codedefinition ausgeführt. Der integrierte Assembler lässt das auf einfache Weise zu. Auch die Berechnung von Bit- und Byte-Offset nach den angegebenen Beziehungen ist in den in Screen # 7 aufgeführten Worten **bitoffset** und **byteoffset** als Codedefinition ausgeführt. Dies ist deshalb angezeigt, da beide Worte beim Setzen eines jeden Bildpunkts aufgerufen werden. Zur Erläuterung sind beide Worte in Screen # 8 nochmal als High-Level-Worte aufgeführt. Die Einhaltung der zulässigen Grafikkoordinaten wird von beiden Worten **nicht** getestet und ist durch das aufrufende Wort zu sichern. Ein komplettes Byte kann mit Hilfe des Wortes **setresbyte** bearbeitet werden. Das Ziehen von waagrechten Linien kann auf diese Weise beschleunigt werden, wobei aber das Byteraster nicht verlassen werden kann. Die Wahl der betreffenden Koordinaten muss deshalb sehr sorgfältig erfolgen.

Screen # 9 stellt nun die Worte zum Bearbeiten des einzelnen Bildpunkts (Dot) bereit. Das Wort **bitpos** dient zur Markierung eines aus dem Bildwiederholspeicher gelesenen Byte. Das Wort **@maske&byte** ist ein Hilfswort, das ausgehend von der Position des Bildpunkts (xpos, ypos) die erforderliche Maske sowie die Andresse im Bildwiederholspeicher bereitstellt und das betreffende Byte ausserdem liest. Nun lassen sich in den Worten **setdot** und **resdot** die erforderlichen logischen Verknüpfungen (bitweises AND bzw. OR) von Maskierungs- und gelesenen Byte vornehmen, bevor das «bearbeitete» Byte wieder in den Bildwiederholspeicher geschrieben werden kann. Mit dem Wort **setresdot** steht ein auf den letzten beiden Worten aufbauendes Wort bereit, dessen Aktivität durch ein zusätzlich zu übergebendes Flag bestimmt wird. Auch das in Screen # 10 enthaltene, der Umschaltung eines Bildpunkts dienende Wort **switchdot** arbeitet in der geschilderten Weise. Die bitweise XOR-Verknüpfung bewirkt dabei das Rücksetzen eines gesetzten Bildpunkts bzw. das Setzen eines zurückgesetzten Bildpunkts. Weiter befindet sich in Screen # 10 noch das Wort **grtest**, mit dessen Hilfe jeder vierte Bildpunkt auf dem Grafikbildschirm angesprochen wird. Während der Entwicklung dieser Komponente diente dieses Wort im wesentlichen der Laufzeitermittlung einzelner Worte. Der Bildaufbau dauerte beispielsweise mit den in High-Level formulierten Worten **bitoffset** und **byteoffset** ursprünglich 42 Sekunden. Deren Codedefinition reduzierte diese Zeit auf 6 Sekunden.

Diese Laufzeiten sollen einzig und allein den Trend aufzeigen, da sich F83 gegenüber anderen Forth-Versionen nicht gerade durch geringe Laufzeiten auszeichnet. Neben dem Setzen und Rücksetzen eines einzelnen Bildpunkts ist die Darstellung von Linien eine weitere Grundvoraussetzung zur Erstellung von Grafiken. Die Abarbeitung der Geradengleichung

$$y = m\,(x - x1) + y1 \text{ mit } m = \frac{y2 - y1}{x2 - x1}$$

erfordert für jeden Bildpunkt eine Multiplikation und eine Division und ist damit recht aufwendig.

Die Suche nach effizienteren Linienalgorithmen brachte beispielsweise den DDA-Algorithmus (Digital Differential Analyzer) und den Bresenham-Algorithmus hervor [2].

Der Vorteil des DDA-Algorithmus liegt in seiner Einfachheit. Allerdings muss eine (einmalige) Division zur Berechnung des Anstiegs m durchgeführt werden und dieser Wert als Realzahl bzw. als spezielles «Integerformat mit Nachkommastellen» gespeichert werden. Die iterative Berechnung mit einem Anstieg eingeschränkter Genauigkeit (Division) führt zur Akkumulierung des Fehlers.

Der Bresenham-Algorithmus umgeht die Nachteile des DDA-Algorithmus (Ungenauigkeit und Division). Der Preis ist ein wesentlich erhöhter Aufwand. Ausserdem benötigt der Bresenham-Algorithmus eine Reihe von Erweiterungen, denn er funktioniert nur, wenn der Anstieg m positiv und kleiner eins ist. Für andere Steigungen ergeben sich Sonderbehandlungen, die die Vorteile des Bresenham-Algorithmus dann aber einschränken. Für Interessenten sei auf eine Implementierung des Bresenham-Algorithmus in [3] verwiesen.

Da Forth von Hause aus die gemischtgenaue Operation */ bereitstellt, wurde trotzdem von der Lösung der Geradengleichung ausgegangen.

Hierbei ist sicherzustellen, dass der Anstieg m endlich ist. Bei den sich anschliessenden Berechnungen wäre sonst eine Division durch null die Folge. Die Darstellung einer senkrechten Linie muss also getrennt erfolgen.

In Screen # 11 werden die beiden zu übergebenden Koordinatenpaare durch das Wort **x1=x2?** auf Gleichheit der x-Koordinate getestet. Das Darstellen einer als Senkrechte erkannten Linie ist dann mit Hilfe des Wortes **senkrechte** vorzunehmen. Weiterhin ist in Screen # 11 noch das Hilfswort **bis<von?** enthalten, das die richtige Reihenfolge der Indizes für die DO ... LOOPs der darstellenden Worte sicherstellen soll. Alle darstellenden Worte enthalten eine solche DO ... LOOP mit einem Schleifenindex von +1 und arbeiten nur dann exakt, wenn der «von-Index» kleiner als der «bis-Index» ist. Andernfalls sind die Koordinatenpaare nur zu vertauschen. Für die Linie selbst ist es ja ohnehin gleichgültig, ob sie von (x1, y1) nach (x2, y2) gezeichnet wird oder umgekehrt.

A,1

```
     0
 0 \ Hercules-Grafik   HERCULES.BLK                    ck 24jul89
 1
 2 Das File HERCULES.BLK ermoeglicht grafische Darstellungen mit
 3 der Hercules-Karte.
 4
 5 Neben dem Setzen einzelner Bildpunkte ist die Ausgabe von Texten deciaal
 6 im Format 43 Zeilen zu je 90 Zeichen moeglich.
 7
 8 Die Verbindung zweier Bildpunkte wird durch Loesung der Geraden-
 9 gleichung vorgenommen. Geschwindigkeitsverbesserungen lassen
10 sich mit dem Bresenham-Algorithmus erreichen.
11
12 Naehere Informationen zu diesem Algorithmus sind bei
13    R. Zech : Forth 83. Muenchen: Franzis Verlag 1987
14 zu finden.
15
```

```
     1
 0 \ Hercules-Interface Loadscreen                     ck 25jul89
 1
 2
 3 cr .( Hercules-Interface wird compiliert...) cr
 4
 5 2 18 thru
 6
 7 cr .( Hercules-Interface ist compiliert.) cr
 8
 9
10
11
12
13
14
15
```

```
     2
 0 \ Zugriff auf gesamten Speicherraum                 ck 26jun89
 1
 2 code l@   ( segment offset -- w )
 3    bx pop  es pop  es: @ [bx] push  next  end-code
 4
 5 code l!   ( w segment offset -- )
 6    bx pop  es pop  es: @ [bx] push  next  end-code
 7
 8 code lc@  ( segment offset -- b )
 9    bx pop  es pop  es: @ [bx] al mov  ah ah sub  1push  end-code
10
11 code lc!  ( b segment offset -- )
12    bx pop  es pop  ax pop  es: al @ [bx] mov  next  end-code
13
14
15
```

```
     10
 0 \ SwitchDot GrTest                                  ck 26jul89
 1
 2 : switchdot   ( xpos ypos  -- )
 3    @maske&byte 3 pick xor -rot 1c! drop ;
 4
 5 : grtest   ( -- )
 6    initgraph
 7    720 0 do
 8       349 0 do j i switchdot 4 +loop
 9    4 +loop
10    resetgraph ;
11
12
13
14
15
```

```
     11
 0 \ x1=x2? senkrechte bis(von?                        ck 25jul89
 1
 2 : x1=x2?   ( x1 y1 x2 y2 -- x1 y1 x2 y2 f )
 3    over 4 pick = ;
 4
 5 : senkrechte   ( x y1 x y2 -- )
 6    rot 2dup < if swap then
 7    swap 1+ swap ?do dup i setdot loop
 8    2drop ;
 9
10 : bis(von?   ( x1 y1 x2 y2 -- x1 y1 x2 y2 f )
11    over 4 pick < ;
12
13
14
15
```

```
     12
 0 \ anstieg !anstieg x_y_tausch 1/anstieg             ck 25jul89
 1
 2 2variable anstieg
 3
 4 : !anstieg   ( x1 y1 x2 y2 -- x1 y1 x2 y2 )
 5    4dup rot - -rot swap - anstieg 2! ;
 6
 7 : x_y_tausch   ( x1 y1 x2 y2 -- y1 x1 y2 x2 )
 8    swap 2swap swap 2swap ;
 9
10 : 1/anstieg   ( -- )
11    anstieg 2@ swap anstieg 2! ;
12
13
14
15
```


A,2

```
     3
 0 \ Hercules-Register Grafik- und Textmodeparameter   ck 26jun89
 1 hex
 2 03b4 constant indexreg
 3 03b5 constant datareg
 4 03b8 constant modectrl
 5 03bf constant configsw
 6 b000 constant seite_0
 7 b800 constant seite_1
 8
 9 create grparaa
10    35 c, 2d c, 2e c, 07 c, 5b c, 02 c,
11    57 c, 57 c, 02 c, 03 c, 00 c, 00 c,
12
13 create txtparaa
14    61 c, 50 c, 52 c, 0f c, 19 c, 06 c,
15    19 c, 19 c, 02 c, 0d c, 0b c, 0c c,
```

```
     4
 0 \ Initialisierung Grafikmodus                       ck 26jun89
 1
 2
 3 : initgraph   ( -- )
 4    82 modectrl pc!
 5    0c 0 do
 6       i indexreg pc!
 7       i grparaa + c@ datareg pc!
 8    loop
 9    03 configsw pc!
10    8a modectrl pc! ;
11
12
13
14
15
```

```
     5
 0 \ Initialisierung Textmodus                         ck 23jun89
 1
 2
 3 : resetgraph   ( -- )
 4    20 modectrl pc!
 5    0c 0 do
 6       i indexreg pc!
 7       i txtparaa + c@ datareg pc!
 8    loop
 9    01 configsw pc!
10    28 modectrl pc! ;
11
12
13
14
15
```

A,2

```
     13
 0 \ loop_ind   y(x)   x(y)                            ck 26jul89

 1
 2 : loop_ind   ( x1 y1 x2 y2 -- y1 x1 x2+1 x1 )
 3    drop 1+ rot dup -rot ;
 4
 5 : y(x)   ( x1 y1 x2 y2 -- )
 6    bis(von? if 2swap then
 7    loop_ind ?do 2dup i swap -
 8       anstieg 2@ */ + i swap  setdot
 9    loop 2drop ;
10
11 : x(y)   ( x1 y1 x2 y2 -- )
12    x_y_tausch 1/anstieg bis(von? if 2swap then
13    loop_ind ?do 2dup i swap -
14       anstieg 2@ */ + i setdot
15    loop 2drop ;
```

```
     14
 0 \ keine_senkrechte line                             ck 25jul89
 1
 2 : keine_senkrechte   ( x1 y1 x2 y2 -- )
 3    !anstieg
 4    anstieg 2@ abs swap abs > if y(x) else x(y) then ;
 5
 6 : line   ( x1 y1 x2 y2 -- )
 7    4 ?enough
 8    x1=x2? if senkrechte else keine_senkrechte then ;
 9
10
11
12
13
14
15
```

```
     15
 0 \ WrChar WrTxt                                      ck 26jun89
 1
 2 hex ffa6 @e 2constant txttable decimal
 3
 4 : wrchar   ( xpos ypos char -- )
 5    3 ?enough 0 max 127 min -rot
 6    8 0 do
 7      2dup i + byteoffset seite_1 swap
 8      txttable 6 pick 8 * i + + lc@ -rot lc!
 9    loop 2drop drop ;
10
11 : wrtxt   ( xpos ypos addr count -- )  \ String im Type-Format
12    4 ?enough  swap 3 roll 3 roll 3 roll  0 do
13       2dup 4 pick i + c@ wrchar
14       swap 8 + swap
15    loop 2drop drop ;
```

```
6
\ Grafikbildschirm loeschen                              ck 26jul89


code cleargraph   ( -- )
  es push   di push
  seite_1 # ax mov
  ax es mov
  di di xor
  4000 # cx mov
  ax ax xor
  repz w stos
  di pop   es pop
  next end-code
```

```
7
\ Byteoffset Bitoffset                                   ck 26jun89

code bitoffset   ( xpos -- offset )
  ax pop   8 # cx mov   cl div
  8 # dx mov   ah al xchg   0 # ah mov   ax dx sub   dx push
  next end-code

code byteoffset   ( xpos ypos -- offset )
  ax pop    4 # cx mov      cl div    ax dx mov
           5a # cx mov      cl mul    ax bx mov
  dx ax mov   al ah xchg   0 # ah mov
         2000 # cx mov      cx mul    ax bx add
  ax pop    8 # cx mov      cl div    0 # ah mov   ax bx add
  bx push
  next end-code
```

```
8
\ Byteoffset Bitoffset SetResByte                        ck 26jun89


\ : byteoffset   ( xpos ypos -- offset )
\     dup 2000 swap 4 mod *
\     swap 4 / 5a * +
\     swap 8 / + ;

\ : bitoffset   ( xpos -- offset )
\     8 mod 8 swap - ;

: setresbyte   ( xpos ypos flag -- )
    0= if 00 else 0ff then
   -rot byteoffset seite_1 swap lc!  ;
```

```
16
\ esc ff (print) init_gra_printer beeps                 ck 26jun89
hex
1b constant esc

: ff   ( -- )   0c emit #out off #line off ;

\ code (print)
\     dx pop   5 # ah mov   21 int    next end-code

: init_gra_printer
   esc emit ascii l emit 00 emit   \ Linker Rand
   esc emit ascii 3 emit 18 emit   \ Zeilenvorschub
   crlf ;

: beeps   ( u -- )
   1 ?enough 0 ?do beep loop ;
```

```
17
\ (hardcopy)                                            ck 26jul89
decimal

: (hardcopy)   ( -- )
   initgraph
   713 0 do
      esc emit ascii K emit 92 emit 1 emit
      0 347 do j i byteoffset seite_1 swap lc@ emit -1 +loop
      crlf
   8 +loop
   crlf  ff
   resetgraph ;
```

```
18
\ hardcopy                                              ck 26jul89

: hardcopy   ( -- )
   3 beeps cr ." Achtung: Druckdauer ca. 2.5 Minuten ! " cr
   ." Soll gedruckt werden ? (j/n) "
   key dup ascii J = swap ascii j = or not if exit then cr
   ." Neues Blatt einlegen, wenn Drucker fertig   <RETURN>" cr
   key drop
   ['] (print) is emit
   init_gra_printer
   (hardcopy)
   ['] (emit) is emit
   5 beeps ." Hardcopy fertig." cr ;
```



```
9
\ Bitpos @maske&byte SetDot ResDot SetResDot             ck 26jul89
create bitpos
   0 c, 1 c, 2 c, 4 c, 8 c, 10 c, 20 c, 40 c, 80 c,

: @maske&byte   ( xpos ypos -- 8b1 adr offset 8b2 )
   over bitoffset bitpos + c@ -rot
   byteoffset seite_1 swap 2dup lc@ ;

: setdot   ( xpos ypos  -- )
   @maske&byte 3 pick or -rot lc! drop ;

: resdot   ( xpos ypos -- )
   @maske&byte 3 pick 0ff swap - and -rot lc! drop ;

: setresdot   ( xpos ypos flag -- )
   if setdot else resdot then ;
```

Listing 1 Die Grafikerweiterung der Hercules-Karte unter Forth. Die Bestandteile der Komponente sind in den Screens 0 ... 18 zu finden.

```
0
\ DEMO.BLK                                               ck 26jul89


Mit diesem kleinen Demoprogramm werden anhand

   der Ausgabe von Texten im Grafikmodus,
   dem Zeichnen von Linien fuer Umrahmungen und
      Funktionsdarstellungen  sowie
   der Hardcopy-Funktion

die Grundfunktionen einer jeden Pixelgrafik fuer die
Hercules-Karte vorgefuehrt.
```

```
1
\ Grafikdemo                                             ck 25jul89

: rahmen   ( -- )
     0   0 719   0 line  719   0 719 347 line
   719 347   0 347 line    0 347   0   0 line ;

: text1   ( -- )
   " Das ist im Grafikmodus der Herculeskarte ausgegebener Text"
;
: text2   ( -- )
   " Ein Pseudo-Rauschvorgang zur Darstellung eines Funktionsver
laufes." ;

: text3   ( -- )
   " In den Textmodus mit beliebiger Taste..." ;
-->
```

```
2
\ seed random random-test                                ck 25jul89

variable seed      1234 seed !

: random   ( n1 -- n2 )
   seed @ 259 * 3 + 32767 and dup seed !
   32767 */ ;

: random-test   ( -- )
   0 0
   720 0 do i 147 random 100 + 2swap 2over line 4 +loop
   2drop ;



-->
```

```
3
\  demo                                                  ck 25jul89

: demo   ( -- )
   initgraph cleargraph
   rahmen
   150 30 text1 wrtxt
   120 250 text2 wrtxt
   random-test
   200 330 text3 wrtxt key drop
   resetgraph ;
```

Listing 2 Demonstrationsprogramm.

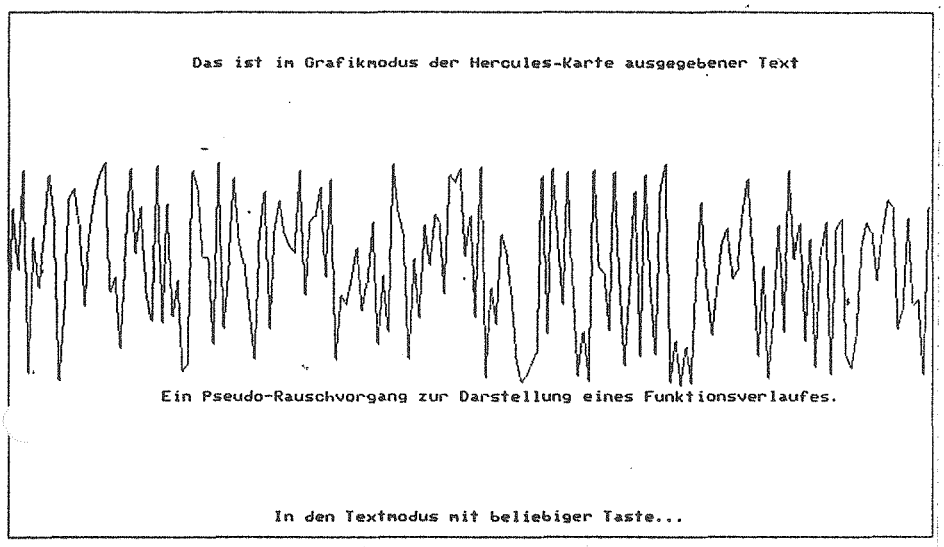

Bild 1 Ausdruck Demonstrationsprogramm.

Screen # 12 beinhaltet die Variable **anstieg.** Diese Variable wird als Bruch bereitgestellt, weshalb sie auch als 32-Bit-Variable definiert wurde. Berechnet wird der Wert des Anstiegs im Wort **!anstieg,** wo auch gleichzeitig die Variable **anstieg** initialisiert wird. Die Worte **x_y_tausch** und **1/anstieg** vertauschen die xy-Koordination bzw. Zähler und Nenner der Variablen **anstieg.**

Um eine geschlossene Linienführung auf dem Bildschirm zu erhalten, werden Geraden mit einem Anstieg $|m| \leq 1$ in der bereits angegebenen Form durch das in Screen # 13 enthaltene Wort **y(x)** dargestellt. Geraden mit einem Anstieg $|m| > 1$ würden bei dieser Vorgehensweise nur noch lückenhaft dargestellt werden. Hier wird dann zur Umkehrfunktion gegriffen. Die Umstellung der Geradengleichung ergibt dann

$$x = \frac{1}{m}(y - y1) + x1$$

Es ist hiernach nichts anderes zu tun, als x- und y-Koordinaten zu vertauschen, Zähler und Nenner des Anstiegs m zu vertauschen und beim Setzen der Bildpunkte wiederum x- und y-Koordinate zu vertauschen. Das Wort **x(y)** erledigt die Darstellung einer Linie mit einem Anstieg $|m| > 1$ unter Nutzung der vor dem definierten Worte **x_y_tausch** und **1/anstieg.** Zusammengefasst wird die Darstellung von Linien mit beliebigem, aber endlichem Anstieg im Wort **keine_senkrechte** (Screen # 14). Als eigentliches Anwenderwort fungiert schliesslich das Wort **line,** das zusätzlich die Zahl der übergebenen Parameter überprüft.

Zur Realisierung von Textausgaben im Grafikmodus wurde der IBM-Zeichensatz des Betriebssystems verwendet. Die auszugebenden Zeichen werden aus jeweils acht Byte gebildet, die nacheinander in einer an der Adresse **txttable** beginnenden Tabelle stehen. Zur Darstellung von Einzelzeichen dient das ebenfalls in Screen # 15 enthaltene Wort **wrchar.** Die betreffenden acht Byte werden aus der Tabelle gelesen und an die gewünschte Stelle im Bildwiederholspeicher geschrieben. Die Ausgabe ganzer Strings geschieht mit Hilfe des Wortes **wrtxt,** das neben den xy-Koordinaten für den Beginn des Wortes auch noch die Parameter eines im TYPE-Format vorliegenden Strings (Adresse, Count) erwartet.

Die weiteren Bestandteile der Komponente dienen der Erstellung einer Hardcopy auf einem grafikfähigen, Epson-kompatiblen Matrixdrucker. In Screen # 16 wird das zur Programmierung des Drucks ständig erforderliche ESC als Konstante **esc** bereitgestellt. Ein Form-Feed wird mit Hilfe des Wortes **ff** definiert (Die Hexadezimalzahl FFh ist zur Unterscheidung hiervon als 0ff einzugeben!). Schliesslich ist in Screen # 16 als Kommentar die Codedefinition des Wortes **(print)** angegeben. Zur Zeichenausgabe über den Drucker wird der betreffende Software-Interrupt verwendet. Dieses Wort wird hier nicht benötigt, da ein solches im F83 bereits enthalten ist. Das Wort **init_gra_printer** übernimmt schliesslich das Einrichten des Drucks. Der linke Rand wird auf die Position 8 und der Zeilenvorschub auf 24/216" festgelegt.

In Screen 17 ist mit dem Wort **(hardcopy)** die eigentliche Hardcopy-Routine zu finden. Nach dem Initialisieren des Grafikmodus wird eine ESC-Sequenz an den Drucker geschickt, die diesen in die Grafikbetriebsart schaltet. Das spaltenweise Lesen des Bildwiederholspeichers und die Ausgabe an den Drucker bewirken eine grafische Ausgabe des Bildschirminhalts im Querformat DIN A4.

Da solche Hardcopy-Routinen naturgemäss einige Zeit dauern, wurde mit dem Wort **hardcopy** ein übergeordnetes Wort geschaffen, das einen knappen Bedienerdialog führt, der immer noch einen rechtzeitigen Ausstieg ermöglicht. Hat man sich dennoch für eine Hardcopy entschieden, werden Vektorisierung der Ausgabe und Druckerinitialisierung ebenfalls von diesem Wort übernommen.

5. Ein kleines Demonstrationsprogramm

Die Anwendung der in der beschriebenen Erweiterung HERCULES.BLK definierten Worte soll anhand des kleinen Demoprogramms DEMO.BLK verdeutlicht werden (Listing 2).

Die eingangs genannten und für die Messtechnik relevanten Bestimmungsstücke einer Grafik sind mit der nachladbaren Komponente HERCULES.BLK realisierbar und mit der im Bild 1 gezeigten Hardcopy eindeutig belegt.

Literatur

[1] Kühnel, C.: Flexibel mit Forth. Zur Softwareproblematik in der Messtechnik. Elektronik, München 20/1988, S. 77–80
[2] Hornung, Ch.; Pöpsel, J.: 3-D à la carte. Teil 3: Vom Linienmodell zur Rasterdarstellung. c't, Hannover 7/1989, S. 122–136
[3] Zech, R.: Forth 83. München: Francis-Verlag 1987