Carsten Schiers

Grafikprozessor und Turbo-Pascal

Besitzer aller Rechner mit grafischen Fähigkeiten haben sich sicherlich schon geärgert, wenn sie unter Universal-Betriebssystemen keine Grafik mehr benutzen konnten. Für Benutzer des NDR-Klein-Rechners ändert sich dies von heute an.

Das Betriebssystem CP/M schafft auf Rechnern, die auf der Hardwareseite nur geringe Ähnlichkeit aufweisen müssen, eine einheitliche Benutzeroberfläche, die es ohne Probleme möglich machen soll, Text und Programme auszutauschen. Diesen Vorzug erkauft man sich aber durch eine Uniformität, die bestimmte Vorzüge eines Rechners, wie zum Beispiel eine leistungsfähige Grafik-Karte, außer acht läßt.
Für Benutzer des NDR-Klein-Rechners unter CP/M-Turbo-Pascal wird nun die Möglichkeit beschrieben, die Grafik-Karte aus Turbo-Pascal heraus zu nutzen. Diese Karte arbeitet mit einem EF9366 der Firma Thomson, der unter den Portadressen 70 bis 7F angesprochen wird. Zusätzlich hat man noch die Möglichkeit, zwischen vier Grafikspeicherseiten hin- und herzuschalten. Benutzer anderer CP/M-Systeme mit diesem Grafikprozessor können die Routinen durch Verändern der Portadressen an ihr System anpassen. Genauere Informationen über die verwendete Hardware finden Sie in [1].

Turbo-Pascal und Maschinencode

Die Sprache Pascal erhebt den Anspruch, logisch strukturiert und selbstdokumentierend zu sein. Die Geschwindigkeit, bei der man nicht mehr zusehen kann, wie der Rechner einen Punkt nach dem anderen setzt, kann man aber im allgemeinen nur mit Maschinensprache erreichen. Die hier präsentierten Routinen vereinigen hoffentlich beides. In Turbo-Pascal ist es nämlich möglich, mit der Prozedur „INLINE" direkt in „Z80" zu codieren. Der dadurch entstehenden Unübersichtlichkeit kann man begegnen, indem man Z80-Programmzeilen in eine Include-Datei verbannt. Diese Include-Datei enthält Pascal-Source-Code und wird während des Compilierens von der Diskette mitübersetzt. Vom Code bekommt man dabei nichts mehr zu sehen. Dieses Verfahren spart zusätzlich auch noch Quelltextspeicher. Wie man Include-Dateien übersetzt, geht aus [2] hervor.
Um die Selbstdokumentierbarkeit zu unterstützen, stehen dem Benutzer der Prozeduren auch noch einige vordefinierte Konstanten und ein vordefinierter Typ zur Verfügung. Dies ermöglicht Konstruktionen, wie gdpout (clearscreen), zur Ausgabe des Control-Codes zum Bildschirmlöschen oder vector (dashed), um die Vektorausgabe gestrichelt einzustellen.

Die Unterprogramme

Bei allen Prozeduren wurde auf die Überprüfung der Parameter aus Geschwindigkeitsgründen verzichtet. Die zulässigen Grenzen werden aber in der Beschreibung angegeben. Es bieten sich

```
const
  selectpen        = 0;
  selecteraser     = 1;
  down             = 2;
  up               = 3;
  clearscreen      = 4;
  home             = 5;
  clearandhome     = 6;
  bigblock         = 10;
  litleblock       = 11;
  continuous       = 0;
  dotted           = 1;
  dashed           = 2;
  dotteddashed     = 3;
  horizontal       = 0;
  vertical         = 8;
  normal           = 0;
  cursiv           = 4;

type
  zeile    = string[80];
procedure waitgdp;
begin
  inline($DB/$70/$E6/$04/$28/$FA);
end;
procedure waitsync;
begin
  inline($db/$70/$02/$04/$28/$fa);
end;
procedure gdpout(wert:byte);
begin
  waitgdp;
  inline($3a/wert/$d3/$70);
end;
procedure vector(vectortype:byte);
begin
  waitgdp;
  inline($db/$72/$e6/$0c/$47/$3a/vectortype/$b0/$d3/$72);
end;
procedure textkind(direction,kind:byte);
begin
  waitgdp;
  inline($db/$72/$e6/$03/$47/$3a/direction/$b0/$47/$3a/kind/$b0/$d3/$72);
end;
procedure textsize(xexpand,yexpand:byte);
begin
 waitgdp;
 inline($3a/xexpand/$47);
 inline($3a/yexpand/$cb/$27/$cb/$27/$cb/$27/$cb/$27);
 inline($b0/$d3/$73);
end;
```

Bild 1. Grafik beim NDR-Klein-Computer von Pascal aus ansprechen

```
procedure textout(textaus:zeile);
var
 i : byte;
begin
 for i:=1 to length(textaus) do gdpout(ord(textaus[i]));
end;
procedure sideselect(leseseite,schreibseite:byte);
begin
 inline($3a/leseseite/$47/$cb/$20/$cb/$20/$cb/$20/$cb/$20);
 inline($3a/schreibseite/$cb/$27/$cb/$27/$cb/$27/$cb/$27/$cb/$27/$cb/$27);
 inline($b0/$d3/$60);
end;

procedure moveto(xkoord,ykoord:integer);
begin
 waitgdp;
 inline($21/xkoord/$7e/$d3/$79/$23/$7e/$d3/$78);
 inline($21/ykoord/$7e/$d3/$7b/$23/$7e/$d3/$7a);
end;
procedure drawto(x2,y2:integer);
var
 x1,y1        : integer;
 xd,yd        : integer;
 befehl       : byte;
begin
 waitgdp;
 inline($21/befehl/$36/$11);
 inline($db/$7a/$67/$db/$7b/$6f/$22/y1);
 inline($db/$78/$67/$db/$79/$6f/$22/x1);
 xd:=x2-x1;
 if xd<0 then
   begin
     inline($21/befehl/$cb/$ce);
     xd:=-xd;
   end;
 yd:=y2-y1;
 if yd<0 then
   begin
     inline($21/befehl/$cb/$d6);
     yd:=-yd;
   end;
 if ((xd>255)or(yd>255)) then
   begin
     inline($2a/xd/$cb/$1c/$cb/$1d/$7d/$d3/$75);
     inline($3a/yd/$cb/$3f/$d3/$77);
     waitgdp;
     inline($3a/befehl/$d3/$70);
   end else
     begin
       inline($3a/xd/$d3/$75);
       inline($3a/yd/$d3/$77);
     end;
   waitgdp;
   inline($3a/befehl/$d3/$70);
end;

procedure plot(x,y:integer);
begin
  moveto(x,y);
  drawto(x,y);
end;

procedure rechteck(x1,y1,x2,y2:integer);
begin
  moveto(x1,y1);
  drawto(x2,y1);
  drawto(x2,y2);
  drawto(x1,y2);
  drawto(x1,y1);
end;
procedure kreis(x0,y0,r:integer);
var
  x,rq,y,u1,u2,u3,u4,v1,v2,v3,v4 : integer;
begin
  rq:=r*r;
  for y:=0 to trunc(r/1.414) do
    begin
      x:=trunc(sqrt(rq-y*y));
      u1:=trunc(x0-x*1.66);   u2:=trunc(x0+x*1.66);   v1:=y0-y;   v2:=y0+y;
      u3:=trunc(x0-y*1.66);   u4:=trunc(x0+y*1.66);   v3:=y0-x;   v4:=y0+x;
      plot(u1,v1);
      plot(u1,v2);
      plot(u2,v1);
      plot(u2,v2);
      plot(u3,v3);
      plot(u3,v4);
      plot(u4,v3);
      plot(u4,v4);
    end;
end;
```

noch einige Möglichkeiten, die durch dieses Prozedurenpaket nicht abgedeckt wurden.

waitgdp():
wartet, bis der Grafikprozessor mit der Abarbeitung des letzten Befehls fertig ist. Die Prozedur kehrt erst dann zurück.

waitsync():
Um flimmerfreie Grafiken zu erzeugen, kann man mit dieser Prozedur auf das alle 20 ms auftretende vertikale Synchronsignal warten, bevor man einen Bildschirmspeicherseitenwechsel vornimmt.

gdpout(wert):
gibt wert an den GDP weiter. Die Prozedur wartet vor der Ausgabe, bis der GDP aufnahmebereit ist. Für diese Prozedur stehen die Konstanten selectpen bis littleblock zur Verfügung.

vector(art):
Der GDP bietet die Möglichkeit, Vektoren auf vier verschiedene Arten zu zeichnen. Mit dieser Prozedur kann man die Linienart einstellen. Als Argument sind die vordefinierten Konstanten „continuous" bis „dotteddashed" empfehlenswert.

textkind(richtung,art):
Bei der Ausgabe von Buchstaben kann man die Richtung (horizontal/vertikal) und die Art (normal/kursiv) der Darstellung wählen. Die restlichen Konstanten helfen auch hier, den Pascal-Code selbstdokumentierender zu gestalten.

textsize(x-size,y-size):
Bei der Ausgabe von Buchstaben kann man diese in X- und Y-Richtung mit verschiedenen Faktoren vergrößern. Die gewohnte 80-Zeichen/Zeile-Darstellung ist dabei die mit dem Faktor 1. Sechzehn verschiedene Faktoren (16 = 0) sind möglich.

textout(string):
Textausgabe ist für einen Grafikprozessor manchmal eine schwierige Angelegenheit. Das BIOS für den NDR-Klein-Rechner löscht, um zu scrollen, eine Seite, und beschreibt sie vollkommen neu. Deswegen ist dieser Vorgang auch relativ langsam. Die Darstellung von Buchstaben erfolgt über die Bildschirmspeicherseiten 0 und 1. Aus diesem Grund empfiehlt es sich, keine Textausgaben mit WRITE oder WRITELN zu machen, sondern textout zu verwenden. Als Parameter kann ein bis zu 80 Zeichen langer String übergeben werden.

sideselect(leseseite,Schreibseite):
gibt die Möglichkeit, zwischen den Bildschirmspeicherseiten 0 bis 3 hin- und herzuschalten. Dabei kann die Seite, die gerade gezeigt wird, verschieden von der sein, auf die der GDP gerade schreibend zugreift.

moveto(x,y):
Diese Prozedur setzt den Schreiber/Löscher auf die Koordinate(x,y). Die x-Koordinate darf dabei im Bereich 0...511, die y-Koordinate im Bereich 1...255 liegen.
drawto(x,y):
Diese Prozedur zeichnet von der mit moveto voreingestellten bis zur angegebenen Koordinate einen Vektor in der mit vektor(art) angewählten Art. Die Parameter müssen hier in den bei moveto angegebenen Grenzen liegen.
plot(x,y):
setzt einen Punkt bei (x,y).
rechteck(x1,y1,xx2,y2):
zeichnet ein Rechteck, das durch die beiden Eck-Koordinaten (x1,y1) und (x2,y2) festgelegt ist.
kreis(x,y,r):
zeichnet einen Kreis mit dem Mittelpunkt bei (x,y) mit dem in y-Richtung exakten Radius r. Da die Bildschirmdarstellung nicht im Verhältnis 1:1 erfolgt, wird der Radius in x-Richtung etwas gestreckt. Das Verfahren ist analog zu dem Basic-Programm aus [3] programmiert. Mit diesen Prozeduren können Sie einiges von dem sehen, was der NDR-Klein-Rechner kann. Turbo-Pascal erzeugt auch relativ schnellen Code. Besonders beim Zeichnen von Kreisen ist aber zu bedenken, daß hier leider die Fließkomma-Rechnung Zeit kostet. Die Implementierung von den aus dem Gundprogramm bekannten Integer Sinus- und Cosinus-Funktionen würde hier sicherlich Abhilfe schaffen.
Die Prozeduren selbst belegen etwa 500 Byte. Bei der Verwendung als Include-File kosten sie kein einziges Byte des Textspeichers.

Literatur

[1] *Klein, Rolf-Dieter:* Mikrocomputer selbstgebaut und programmiert. Franzis-Verlag
[2] *Borland International:* Turbo-Pascal Reference Manual
[3] *Welsch, Norbert:* Kreise einmal anders. mc 1985, Heft 4

8086-/80286-Assembler mit Editor

Nach dem bewährten Konzept von Turbo-Pascal, auch vom Erscheinungsbild her nicht unähnlich und in einer vergleichbaren Preisklasse präsentiert sich ein Editor-Assembler namens TASM für die Prozessoren 8088, 8086, 80186 und 80286. Das Softwarepaket wurde von der wenig bekannten Firma Speedware im kalifornischen Folsom entwickelt und wird hierzulande von Lauer & Wallwitz in Wiesbaden vertrieben, die auch das Handbuch übersetzten; englisch blieben lediglich die Erläuterungen der einzelnen CPU-Mnemonics.
Ein Hauptvorteil von TASM ist die Möglichkeit, ein Quellprogramm zu editieren, zu assemblieren und schließlich sogar zu starten, ohne irgend ein Programm nachladen zu müssen. Ferner läuft die Assemblierung im Schnitt doppelt so schnell wie bei vergleichbaren Intel- oder Microsoft-Assemblern ab.
Nach Aussage des Handbuchs ist die Assembler-Syntax mit der des bekannten Microsoft-MASM weitgehend identisch; Unterschiede beträfen im wesentlichen die 8087-Behandlung, die nicht ganz so umfassend erfolgt wie bei MASM. Ferner werden arithmetische Ausdrücke nach den Pseudo-Anweisungen DD, DQ und DT nicht unterstützt. Auch geschachtelte Makros (Makros, die ihrerseits Makros aufrufen) sind verboten. Unterschiede gibt es weiterhin bei Bitvergleichen, da MASM intern mit einem 17-Bit- und TASM mit einem 16-Bit-Format arbeitet. Kleinere Unterschiede betreffen noch die Listing-Ausgabe, sind jedoch ohne Bedeutung.
Beim Test (Version 1.03B) stellten sich jedoch weitergehende Unterschiede heraus. So ist bei MASM zwischen dem zu einem Label gehörenden Doppelpunkt (z. B. START:) und dem darauf folgenden Mnemonic kein Leerraum erforderlich, während TASM in diesem Fall das Ende des Label-Namens erst beim nächsten Leerraum (also hier hinter dem Befehl!) erkennt. Ähnlich verhält sich TASM auch beim Gleichheitszeichen für Wertzuweisungen: Vor ihm muß im Gegensatz zum Microsoft-Assembler ein Leerraum stehen.
Auch trat beim Assemblieren eines etwa 16 KByte umfassenden Quelltextes, der unter MASM ohne Fehler läuft, laufend die Fehlermeldung „Phase Error between Passes" auf – ein Zeichen dafür, daß sich der Assembler irgendwo im erzeugten Code verzählt hat. Aber wo und warum, ließ sich beim besten Willen nicht ermitteln. (Nebenbei, die uns vorliegende MASM-Version tut dies auch manchmal, zum Beispiel bei Verwendung des Befehls TEST Memory, Immediate.)
Beim Start meldet sich TASM ähnlich wie Turbo-Pascal mit einem Menü. Im Gegensatz zu Turbo-Pascal kann man beim Laden des Editor-Assemblers schon den Filenamen einer Quelldatei angeben, die dann automatisch nachgeladen wird. Die Directory-Funktion des Menüs zeigt unüblicherweise auch die Systemdateien MSDOS.SYS und IO.SYS an – dies mag man als Vor- oder Nachteil werten...
Der integrierte Editor weist wie das gesamte Konzept große Ähnlichkeiten mit dem Turbo-Pascal-Editor auf – ist aber leider viel langsamer. Drückt man die PGUP-Taste, obwohl der Cursor schon ganz oben steht, so baut der Editor trotzdem den Bildschirm komplett neu auf, was sich darin äußert, daß der Cursor 1,3 Sekunden lang über den Schirm huscht. Bei Verwendung der Pfeiltasten oben/unten überholt die Tastatur-Repeat-Funktion in kürzester Zeit die Scroll-Geschwindigkeit, was sich in einem sekundenlangen Weiterlaufen des Cursors nach dem Loslassen der Taste äußert. Möchte man in einer längeren Quelldatei einen häufig vorkommenden String durch einen anderen austauschen, so kann man derweil ohne weiteres eine Tasse Kaffee holen gehen.
Für den Assemblier-Vorgang stehen zahlreiche Optionen zur Verfügung: Source und Objekt wahlweise aus dem Speicher oder von Disk holen, COM-, EXE- oder linkbare OBJ-Dateien erzeugen, Listing-Ausgabe unterdrücken oder auf Bildschirm, Drucker oder Disk-Datei und so weiter. Möchte man die Bildschirm-Ausgabe beim Assemblieren anhalten, so drückt man ganz instinktiv meist CTRL-Numlock. Mit einer anderen Taste geht's dann weiter. Mit dem deutschen Tastaturtreiber KEYBGR.COM führte dies aber dazu, daß TASM anschließend überhaupt nicht mehr auf die Tastatur reagierte. Zum Glück gibt es die Alternative, eine Listing-Ausgabe mit der Leertaste anzuhalten und weiterlaufen zu lassen.
Das mitgelieferte Install-Programm ist im Vergleich zum sonstigen Komfort des Software-Pakets leider sehr rudimentär. Es erlaubt die Voreinstellung von Assembler-Optionen, die Angabe des verwendeten Bildschirm-Typs und die Zuweisung von Tastenkombinationen für den Editor. Dabei ist die vorher vorhandene Einstellung nicht zu sehen, und wenn man eine Taste installiert hat, erhält man keinerlei brauchbare Rückmeldung. Bei Auslieferung entspricht die Tastenbelegung übrigens etwa der von Wordstar (CTRL-Q F für Suchen, CTRL-K D zum Verlassen des Editors, wofür aber auch ESC funktioniert). Fe.