

DOS EXTRA

18,- DM 145 Ös 18,- sfr

Sonderausgabe DOS-EXTRA

Nr. 5 1988/89

ISSN 0934 - 2842

DMV-Verlag

Eine Publikation der
DOS INTERNATIONAL

Grafikkarten von Hercules bis VGA

- Aufbau
- Funktionsweise
- Programmierung

BIOS

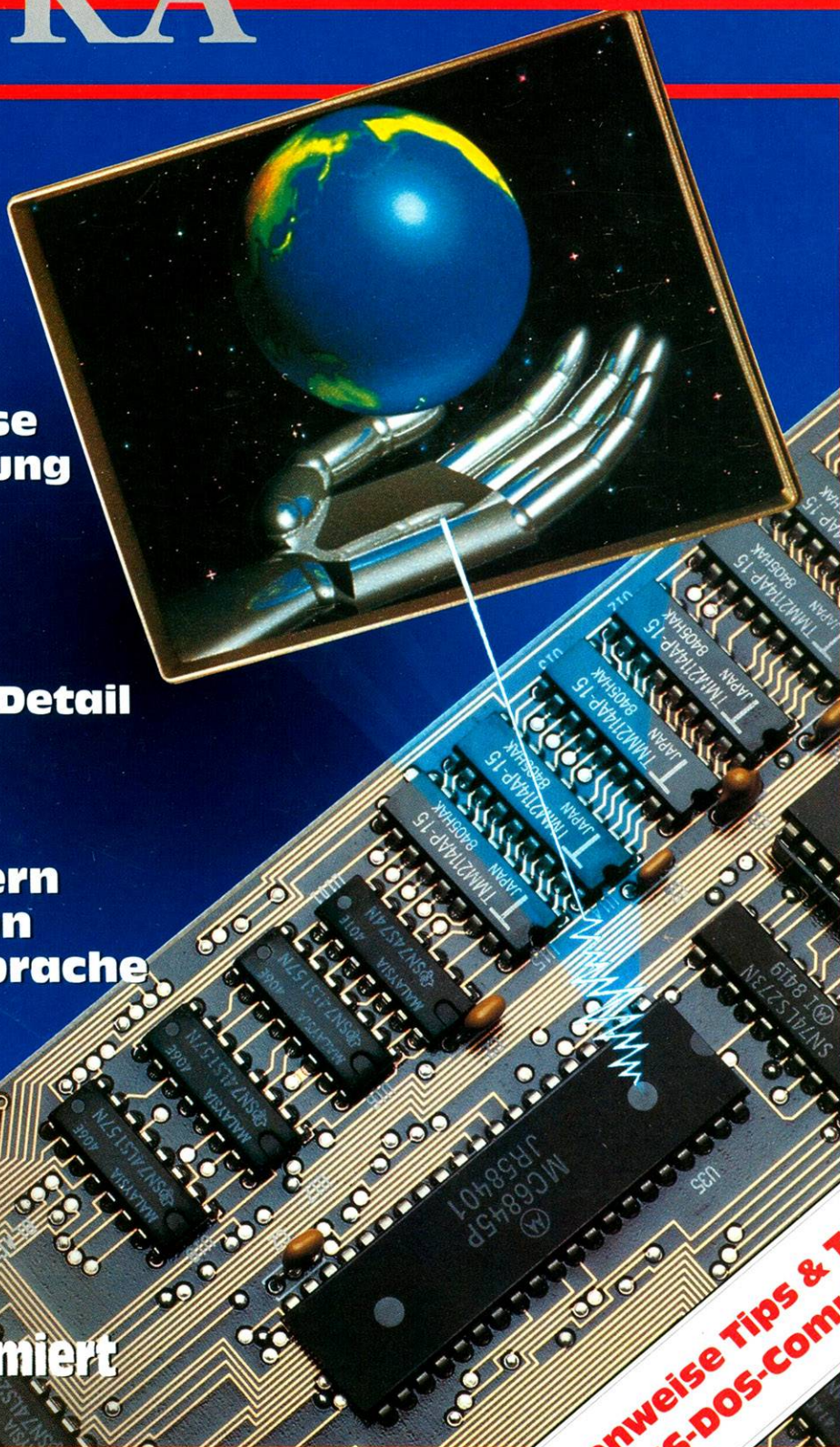
- Grundlagen
- Interrupts im Detail

Basic

- GW-Basic intern
- Einbinden von
Maschinensprache

Utilities für jeden Zweck

Gerätetreiber selbst programmiert



**Seitenweise Tips & Tricks
für MS-DOS-Computer**

Grenzenlose Vielfalt für Ihren PC



Die Anzahl der Einsatzgebiete für PCs ist schon sehr groß, und jeden Tag kommen neue hinzu – in letzter Zeit speziell im Bereich der Verarbeitung von Texten und Grafiken, die hohe Bildauflösung erfordern. Dieser Entwicklung haben wir mit der fünften Ausgabe des DOS Extra Rechnung getragen: Ob Sie nun in Basic oder Assembler programmieren, Grafikutilities oder Gerätetreiber entwickeln möchten, leistungsfähige Programme zum Abtippen oder kurze Tips und Tricks zum sofortigen Ausprobieren suchen, in dieser Ausgabe des DOS Extra ist für jeden etwas dabei.

Für alle Programmierer von höchstem Interesse ist der große BIOS-Schwerpunkt, der Ihnen, neben wichtigen Grundlagen, auch detaillierte Informationen zu jedem einzelnen Interrupt und dessen Funktionen zur Verfügung stellt. Basic-Programmierer erfahren in einem Grundlagenbeitrag den Aufbau eines Basic-Programms auf Datenträgern und im Arbeitsspeicher. Auch ein Verfahren zum Einbinden von Maschinenspracheroutinen in eigene Basic-Programme wird vorgestellt.

Grafik ist ein weiteres Schwerpunktthema dieser Ausgabe. Im fünften DOS Extra geht es um die Grundlagen der Grafikkarten-Programmierung in Assembler. Gleichgültig, ob Sie eine Hercules-, CGA-, EGA- oder bereits eine VGA-Karte besitzen. Sie erhalten zu jedem Kartentyp die erforderlichen Daten und Routinen. Wie sich Grafiken in 16 Farben auf der CGA-Karte darstellen lassen, wird ebenfalls gezeigt.

Weitere ausführliche und tiefgehende Informationen für PC-Besitzer liefert ein Beitrag zu dem – zugegebenermaßen nicht ganz einfachen – Thema Gerätetreiber. Doch

mit den reichhaltigen Informationen zu diesem Themenkomplex werden auch Sie zum Fachmann auf diesem Gebiet.

Natürlich ist in dieser Ausgabe auch wieder eine eigene Rubrik für Tips und Tricks vorhanden, die Sie mit vielen Hinweisen und Kniffen zur Bedienung und Programmierung Ihres PC versorgt. Und wenn Sie ein gutes Programm zum Abtippen suchen – wie wäre es mit einem komfortablen Editor in GW-Basic oder einem flotten Basic-Spiel? Mehrere Utilities runden dieses DOS Extra ab. Mit dem speicherresidenten Programm »snapshot« kopieren Sie den Inhalt des Bildschirms in eine ausführbare Datei (für alle Grafikkarten geeignet), mit »palette« ändern Sie nachträglich deren Farben und mit »cmosview« lesen und beschreiben Sie den CMOS-RAM eines AT oder 386er PC, der die Setup-Informationen für die Systemkonfiguration enthält.

Weitere Utilities manipulieren Druckerzeichensätze und suchen auf Ihrer Festplatte nach den gewünschten Dateien.

Wie Sie sehen, erwartet Sie auf den folgenden Seiten wieder jede Menge interessanter Lesestoff, der Ihnen viele unterhaltsame Stunden mit dem PC garantiert. Viel Vergnügen!

Ihr

Martin Aschoff

Basic:

GW-Basic öffnet die Fenster	12
– Befehlssatzerweiterung selbst gemacht	
GW-Basic intern	18
– Das Format der Basic-Programme	
Tips und Tricks für Basic	20

Grundlagen:

Der Boot-Record	22
16 Farben für die CGA-Karte	23
Die MS-DOS-Stapeldateien	27

BIOS:

Das BIOS: einfach unersetzlich	32
– Grundlagen	
Klare Sicht mit dem Video-Interrupt 10hex	38
Der Disketten- und Festplatten-Interrupt 13hex	42
Am Draht zur Welt:	
Schnittstellen-Interrupt 14hex	44
Antiquität: Kassetten-Interrupt 15hex	45
Immer am Drucker: Tastatur-Interrupt 16hex	47
Der PC macht Druck: Drucker-Interrupt 17hex	49
Zeit im Griff: Systemuhr-Interrupt 1Ahex	49

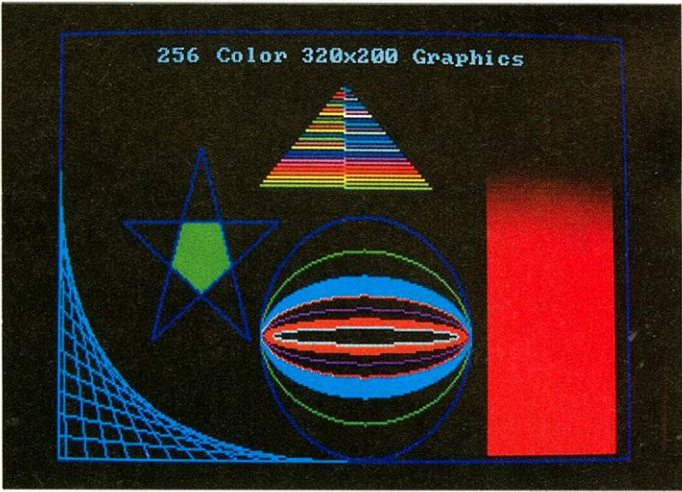
Grafik:

Den Bildschirmkarten aufs Register geschaut	50
– Grundlagen	
Hercules – Der Held der Grafikwelt	54
Die bunte Welt der CGA	60
EGA: Farbvielfalt wirkt elegant	63
VGA – das höchste der Grafikgefühle	65



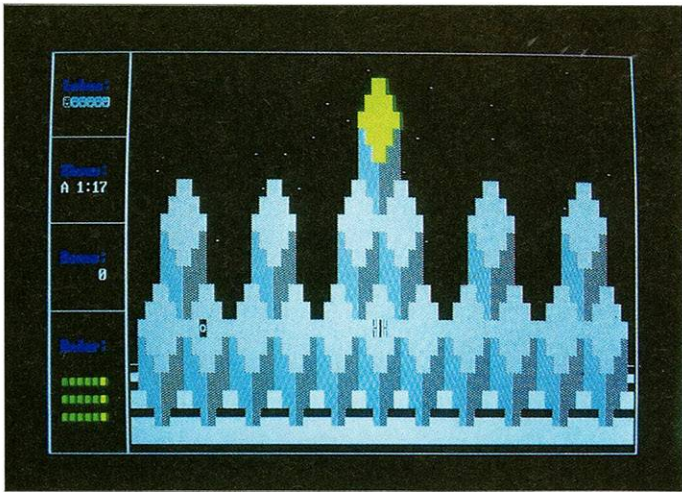
Für die Steuerung eines Computers ist das BIOS zuständig. Wie man es direkt anspricht und wie Sie aus Ihrem PC deutlich mehr »herausholen« können, finden Sie ab

Seite 32



Fünf verschiedene Grafikkarten haben sich beim PC als Standard durchgesetzt. Wie man mit ihnen arbeitet, wie man sie programmiert und welche Vorteile sie bieten, lesen Sie ab

Seite 50



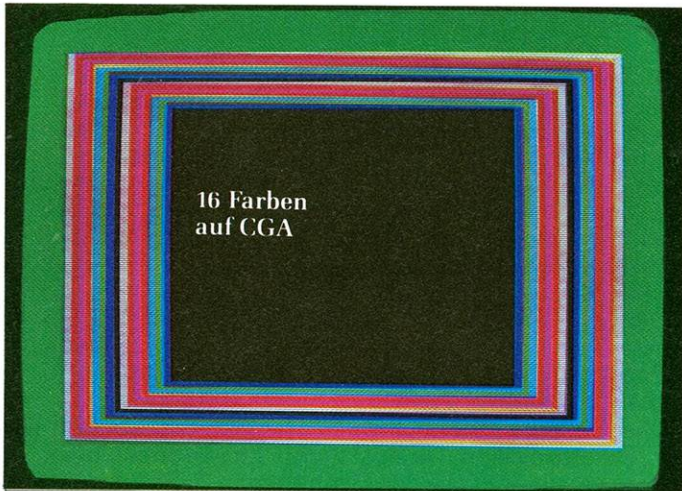
Unendliche Weiten warten auf Sie, wenn Sie das Raumschiff »Stardos« besteigen. Dringen Sie vor in die Tiefen des Raums und verteidigen Sie Ihr Schiff gegen alle Gefahren, die solch eine Expedition begleiten.

Seite 92



Was hilft ein PC, der nicht mit seiner Umwelt kommunizieren kann? Wie man sich die dazu notwendigen Gerätetreiber selbst schreibt, lesen Sie ab

Seite 153



16 Farben auf der CGA-Karte – dazu brauchen Sie keine neue Hardware. Richtig programmiert bringt auch die erste Farbgrafikkarte für PCs erstaunliche Leistungen.

Seite 23



Einen Editor zum Schreiben kleiner Texte, zum Programmieren hilfreicher Batch-Dateien oder zum Eingeben von Programm listings kann man sich selbst in Basic schreiben. Änderungen nach persönlichen Bedürfnissen bleiben damit Tür und Tor geöffnet.

Seite 78

Tips & Tricks:

Tips und Tricks für Programmierer	68
Tips und Tricks für Einsteiger	72
Tips und Tricks zu MS-DOS	74

Tips zur Eingabe:

Checksummer für sichere Listing-Eingabe – Prüfsummenprogramm	6
---	---

Software:

Editor in GW-Basic	78
Raumschiff Stardos – Spiel	92

Utilities:

Mouse-Painting – ein Malprogramm mit professionellem Touch	101
Minimouse – Eine große Sache	104
Wer sucht, der findet – Stringsuche	108
Mehr als ein Blick ins CMOS-RAM – Systemkonfiguration	116
Snapshot: Bildschirmfotos per Software – Bildschirm retten	127
Druckerzeichensatz selbst gemacht	141

Gerätetreiber:

So arbeitet ein Gerätetreiber	153
Kommandos für Gerätetreiber	158
Ein Druckertreiber im Selbstbau	163

Rubriken:

Editorial	3
Impressum	170
Inserenten	170

Checksummer für sichere Listing-Eingabe

Alle Programmlistings werden in der DOS International nach einem speziellen Verfahren mit Prüfsummen und Zeilennummern versehen und abgedruckt. Das für die Überprüfung notwendige Prüfsummenprogramm »Scheck« liegt seit einiger Zeit in einer überarbeiteten und verbesserten Version 2.0 vor. Ein neuer Prüfsummenalgorithmus macht die Überprüfung von abgetippten Listings noch sicherer.

Zunächst einmal einige grundsätzliche Anmerkungen zu Listings in DOS International: Alle abgedruckten Listings sind mit Prüfsummen und Zeilennummern versehen. Jede Zeile beginnt mit einer dreistelligen Prüfsumme, die in eckige Klammern gesetzt ist. Diese Prüfsumme geben Sie beim Abtippen nicht mit ein. Auf die Prüfsumme folgt die Zeilennummer, die mit einem Doppelpunkt abgeschlossen wird. Auch die Zeilennummer geben Sie nicht mit ein. Nach dem Doppelpunkt folgt ein Leerzeichen und darauf dann der eigentliche Text der Programmzeile. Nur diesen letzten Teil der abgedruckten Zeile geben Sie beim Abtippen ein.

Bei allen Programmzeilen wird im Abdruck nach jeweils 55 Zeichen ein Zeilenumbruch vorgenommen. Die umgebrochene Teilzeile erhält dabei weder Prüfsumme noch Zeilennummer. Bei der Eingabe geben Sie umgebrochene Zeilen bitte in einem Stück ein. Betätigen Sie also nicht die Return-Taste, wenn ein Zeilenumbruch erscheint, sondern nur, wenn eine neue Zeilennummer auftaucht.

Als Ausnahme werden Listings in GW-Basic ohne zusätzliche Zeilennummer gedruckt, weil derartige Listings Zeilennummern automatisch als Bestandteil der Sprache enthalten müssen. In diesem Falle erscheint die Zeilennummer ohne Doppelpunkt und muß unbedingt mit abgetippt werden.

Um das Programm in den Computer zu bekommen, können Sie einen beliebigen Texteditor verwenden, der reinen ASCII-Text erzeugen kann. Das Programm ist richtig eingegeben, wenn Sie den Text beim Auflisten mittels Type-Kommando unter MS-DOS korrekt lesen können.

Wie Sie die eingegebenen Programme starten, hängt ganz vom Typ des Programms ab. Für die meisten Listings brauchen Sie einen Compiler, um den Programmtext in Maschinensprache zu übersetzen. Beachten Sie unbedingt die Hinweise in der Programmbeschreibung.

Beim Abtippen von Programmen sind Tippfehler kaum zu vermeiden. Mit »Scheck«, dem Prüfsummen-Programm der DOS International, können Sie die abgetippten Listings aber auf eventuelle Fehler überprüfen. Das Programm ist in Turbo Pascal 3.0 geschrieben und im Quellcode und als ausführbares Programm auf jeder Databox-Diskette enthalten. Die neueste Version 2.0 dieses Checksummers finden Sie im folgenden Listing. Es erlaubt nicht nur die Diagnose von fehlerhaft eingegebenen Zeilen, sondern ermöglicht mit einem integrierten Editor auch gleich die Korrektur solcher fehlerhafter Zeilen.

Scheck wird gestartet durch Eingabe des Dateinamens:

```
scheck <name.ext>
```

Anstelle von »<name.ext>« geben Sie bitte den vollständigen Namen des Listings an, das Sie überprüfen wollen. Zuvor sollten Sie sich sicherheitshalber eine Kopie Ihres Listings anfertigen. Scheck arbeitet zwar nicht mit dem

Original, sondern mit einer temporären Zwischendatei und verändert die Originaldatei erst ganz zum Schluß, nach dem erfolgreichen Abschluß der Arbeit. Dennoch sollte man auch die geringste Möglichkeit eines Datenverlustes etwa durch Fehler beim Speichern ausschalten.

Wie Sie im Bild sehen, zeigt Scheck die aktuelle Zeile stets in einem besonderen Bildschirmfenster an. Die Prüfsumme wird links darunter angezeigt. Beim Überprüfen Ihres Listings vergleichen Sie die angezeigte Prüfsumme mit der in DOS International in eckigen Klammern abgedruckten Prüfsumme. Rechts daneben werden die aktuelle Zeilennummer und der Dateiname angezeigt. Im unteren Fenster werden Fehlermeldungen und Hinweise auf die weitere Bearbeitung ausgegeben. Im abgebildeten Fall können Sie mit [Return] die Zeile übernehmen oder mit [E] den Editor aufrufen.

Der Editor dient der Korrektur fehlerhafter Zeilen. Er arbeitet immer im Einfügemodus, so daß fehlerhafte Eingaben extra gelöscht werden müssen. Sie können also nichts versehentlich überschreiben. Mit den Cursor-Tasten steuern Sie den Cursor im Editierfenster. [Home] bringt ihn an den Anfang der angezeigten Zeile und [End] ans Ende. Löschen können Sie das Zeichen links vom Cursor mit der großen Löschtaaste oberhalb der Return-Taste. [Del] löscht das Zeichen unter dem Cursor. Mit [Return] übernehmen Sie die korrigierte Zeile. Die Leerzeichen zu Anfang einer Zeile werden übrigens bei der Prüfsummenberechnung nicht mit einbezogen. Somit können Sie beispielsweise bei Pascal-Programmen mit beliebigen Einrückungen arbeiten. Mitten in der Zeile vorkommende Leerzeichen sind allerdings für die Prüfsumme relevant. Steuerzeichen wie zum Beispiel Tabulatoren dürfen in einer Zeile überhaupt nicht verwendet werden (und kommen auch in den abgedruckten Listings nicht vor).

Haben Sie so das gesamte Programm oder auch nur »verdächtige« Teile davon durchgearbeitet, so speichert ein letzter Druck auf die Return-Taste das korrigierte Listing. Scheck wird beendet, und das fehlerfreie Programmlisting steht zu Ihrer Verfügung.

Scheck erlaubt aber noch einen zweiten Weg, das Listing korrekt einzugeben. Diesmal geben Sie die Prüfsumme gleich bei der Eingabe des Listings (zum Beispiel mit einer Textverarbeitung) ein. Die eckigen Klammern lassen Sie dabei aber weg, desgleichen die Zeilennummer und den folgenden Doppelpunkt. Hinter der dreistelligen Prüfsumme geben Sie ein Leerzeichen ein, und danach den normalen Programmtext dieser Zeile. Wenn Sie den Text so aufbereitet vorliegen haben, dann können Sie die automatische Überprüfung mit Scheck nutzen. Rufen Sie dazu den Prüfsummer mit der Autoprüf-Option auf:

```
scheck <name.ext> /a
```

Den Schrägstrich »/« können Sie auch durch ein Minuszeichen »-« ersetzen. Desgleichen darf die Option auch als Großbuchstabe erscheinen. Scheck verwandelt das Listing nun in eine Quelldatei für den jeweiligen Compiler, indem die drei Prüfsummenstellen und das Leerzeichen vom Rest der Zeile abgetrennt werden. Aus der so entstandenen Programmzeile berechnet Scheck die Prüfsumme und vergleicht sie mit der zuvor abgetrennten Summe, die Sie eingegeben haben. Falls ein Unterschied auftritt, haben Sie Gelegenheit, die fehlerhafte Zeile manuell zu korrigieren.

Nach einem solchen Durchlauf von Scheck steht wieder das fehlerfreie Programm, allerdings ohne Prüfsumme, auf der Diskette. Und wieder gilt, daß Sie nicht mit dem Original arbeiten sollten, sondern nur mit einer Kopie, um im Zweifelsfall noch einmal von vorne beginnen zu können. Scheck verfügt aber noch über zwei weitere Optionen. Die von Fall zu Fall in der DOS International abgedruckten Hexdump-Listings können Sie mit Scheck im Binär-Modus direkt in lauffähige Programme umwandeln. Dazu

müssen Sie das Hexdump-Listing mit einem ganz normalen Editor (zum Beispiel Turbo Pascal, Wordstar oder Edlin) als Textdatei eingeben. Anschließend starten Sie SCheck zunächst zum Überprüfen des abgetippten Listings ohne weitere Option. Nach dem Kommando

```
scheck <name.ext>
```

wird der Text wie bei jedem anderen Listing auch zeilenweise ausgegeben und kann überprüft werden. Der Editor steht ebenfalls wieder zur Verfügung. Befindet sich das Programm korrekt auf Diskette, so wird SCheck mit

```
scheck <name.ext> /b
```

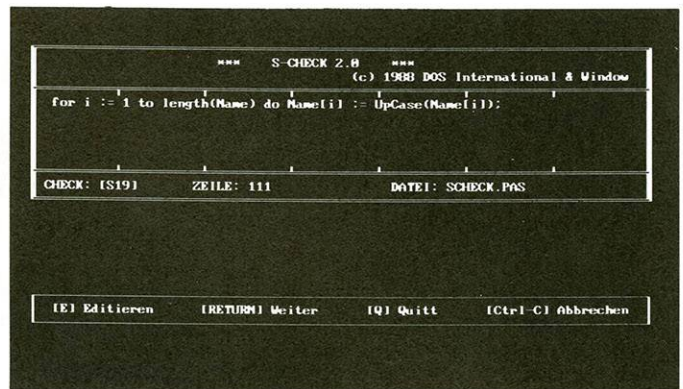
erneut gestartet. Durch den Aufruf mit der B-Option wird SCheck dazu veranlaßt, eine Hexdump-Textdatei in ein ablauffähiges Programm umzuwandeln. SCheck erzeugt in jedem Fall eine Datei mit der Namensweiterung »bin«. Um daraus eine ablauffähige Datei zu machen, müssen Sie sie umbenennen:

```
rename <name>.bin <name>.com
```

Falls eine Exe-Datei erzeugt werden soll (bitte entnehmen Sie dies der jeweiligen Programmbeschreibung) lautet der entsprechende Befehl:

```
rename <name>.bin <name>.exe
```

Eine weitere Option »/1« ermöglicht es, Listings nach der alten Prüfsummenkonvention, also Listings aus den DOS-Ausgaben vor Ausgabe 10'88 oder aus den Extra-Ausga-



Arbeitsbildschirm von SCheck in Aktion

ben 1 bis 4, zu überprüfen und korrekt einzugeben. Wenn Sie SCheck zusätzlich zu den etwa vorhandenen anderen Optionen mit der Option »/1« aufrufen, verhält SCheck 2.0 sich genauso wie sein Vorgänger-Programm. Sie sehen, auch mit SCheck 2.0 ist das korrekte Abtippen von Programmlistings kein bißchen komplizierter geworden – aber durch einen verbesserten Prüfsummenalgorithmus eben noch um einiges sicherer.

(ev)

```
[VR2] 1: program scheck;
[000] 2:
[B7A] 3: { Prüfsummen-Utility für DOS International
      und Window }
[000] 4:
[284] 5: { Volker Everts, 10.1.88 }
[KB3] 6: { Version 2.0 }
[HM4] 7: { Letzte Änderung: 5.8.88 }
[000] 8:
[000] 9:
[6X0] 10: const
[9F9] 11: ZEICHEN : string[32] = '0123456789ABCDEFGH
      HKLMNPRSTUVWXYZ';
[XN1] 12: SPACE = #32;
[BD1] 13: BS = #8;
[4H1] 14: TAB = #9;
[EH1] 15: BEL = #7;
[GD1] 16: CR = #13;
[UP1] 17: EMPTY = '';
[PK1] 18: ESC = #27;
[000] 19:
[ST0] 20: type
[Y84] 21: ZeilenTyp = string[255];
[000] 22:
[RN0] 23: var
[1X2] 24: Zeile: ZeilenTyp;
[BC3] 25: ps, check: string[5];
[Z56] 26: Name, TempName, Option: string[40];
[NU2] 27: Ext: string[4];
[C72] 28: t, temp: text;
[TE2] 29: f: file of byte;
[HY3] 30: Nummer, b, code: integer;
[HH1] 31: by: byte;
[GC1] 32: ch: char;
[N85] 33: weiter, auto, bin, eins: boolean;
[000] 34:
[000] 35:
[7R2] 36: procedure init;
[000] 37:
[RN0] 38: var
[8V1] 39: i: integer;
[000] 40:
[MU0] 41: begin
[F92] 42: auto := false;
[N02] 43: bin := false;
[Z72] 44: eins := false;
[GH2] 45: window(1,1,80,25);
[AW2] 46: ClrScr; writeln;
[XK9] 47: writeln('
      ');
[FP9] 48: writeln('*** S
      -CHECK 2.0 ***
[37B] 49: writeln('
      (c) 1988 DOS International & Windo
      w ');
```

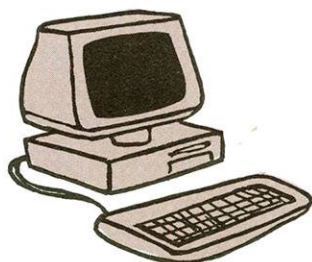
```
[SM9] 50: writeln('
[NS8] 51: writeln('
[NS8] 52: writeln('
[NS8] 53: writeln('
[NS8] 54: writeln('
[NS8] 55: writeln('
[YL9] 56: writeln('
[6BA] 57: writeln(' CHECK: ZEILE:
[VK9] 58: writeln('
[LZ1] 59: GotoXY(1,20);
[ZY7] 60: writeln('
[TR8] 61: writeln('
[EW7] 62: writeln('
[3M3] 63: if ParamCount <> 0 then
[DB4] 64: for i := 1 to ParamCount do
[MU0] 65: begin
[WH3] 66: Option := ParamStr(i);
[BF2] 67: if (Option = '-a')
[VG2] 68: or (Option = '-A')
[3H2] 69: or (Option = '/a')
[7G5] 70: or (Option = '/A') then auto := tru
e
[ZR0] 71: else
[CF2] 72: if (Option = '-b')
[WG2] 73: or (Option = '-B')
[4H2] 74: or (Option = '/b')
[W85] 75: or (Option = '/B') then bin := true
[ZR0] 76: else
[VE2] 77: if (Option = '-1')
[7E5] 78: or (Option = '/1') then eins := tru
e
```

Listing »scheck.pas« in Turbo Pascal 3.0


```
[W93] 79:     else Name := Option;
[4M0] 80:     end
[V53] 81:     else Name := EMPTY;
[P31] 82:     repeat
[C83] 83:     if Name = EMPTY then
[MU0] 84:     begin
[TZ1] 85:         GotoXY(3,21);
[A86] 86:         write('Geben Sie einen Dateinamen e
in');
[F42] 87:         GotoXY(54,12);
[LV1] 88:         BufLen := 24;
[RT1] 89:         read(Name);
[TZ1] 90:         GotoXY(3,21);
[ET3] 91:         write('
');
[ZN0] 92:     end;
[NF2] 93:     assign(t, Name);
[IL3] 94:     {$I-} reset(t); {$I+}
[463] 95:     if IOresult > 0 then
[MU0] 96:     begin
[YF2] 97:         for i := 1 to 3 do
[MU0] 98:         begin
[642] 99:             GotoXY(10,21);
[NH6] 100:            write('+++++ D A T E I F E H L
E R +++++', BEL);
[HN1] 101:            delay(200);
[642] 102:            GotoXY(10,21);
[N54] 103:            write('
');
[HN1] 104:            delay(200);
[ZN0] 105:        end;
[2A2] 106:        Name := EMPTY;
[F42] 107:        GotoXY(54,12);
[4B3] 108:        write('
');
[ZN0] 109:    end;
[V93] 110:    until Name <> EMPTY;
[S19] 111:    for i := 1 to length(Name) do Name[i] :=
UpCase(Name[i]);
[YR2] 112:    i := pos('.',Name);
[HS1] 113:    if i = 0 then
[MU0] 114:    begin
[RP2] 115:        Name := Name + '.';
[AU2] 116:        i := length(Name);
[ZN0] 117:    end;
[5N3] 118:    Ext := copy(Name, i, 4);
[204] 119:    Name := copy(Name, 1, i-1);
[F42] 120:    GotoXY(54,12);
[HU2] 121:    write(Name, Ext);
[GM3] 122:    TempName := Name + '.$$';
[2Y3] 123:    assign(temp, TempName);
[9F2] 124:    rewrite(temp);
[DS1] 125:    if bin then
[MU0] 126:    begin
[TG3] 127:        assign(f, Name + '.BIN');
[YY1] 128:        rewrite(f);
[F92] 129:        auto := false;
[ZN0] 130:    end;
[AV3] 131:    if eins then Nummer := 0
[DR2] 132:    else Nummer := 1;
[ZN0] 133: end;
[000] 134:
[000] 135:
[000] 136:
[P45] 137: function Upper(ch: char): char;
[4X1] 138: var c: char;
[MU0] 139: begin
[LM1] 140:     case ch of
[MW1] 141:         'ä': c := 'Ä';
[0Y1] 142:         'ö': c := 'Ö';
[4Y1] 143:         'ü': c := 'Ü';
[K73] 144:         else c := UpCase(ch);
[ZN0] 145:     end;
[XT1] 146:     Upper := c;
[ZN0] 147: end;
[000] 148:
[000] 149:
[000] 150:
[S36] 151: procedure Pruefe(var z: ZeilenTyp);
[000] 152:
[TYA] 153: { Prüfsumme berechnen. Führende Leerzeichen
und Tabulatoren }
[ZCA] 154: { werden nicht berücksichtigt. Bei Steuerzei
chen im Text }
[3X6] 155: { erfolgt eine Warnung
}
[000] 156:
[CP3] 157: var a, p, i, j: integer;
[000] 158:
[MU0] 159: begin
[1Z0] 160:     p := 0;
[6Y0] 161:     a := 1;
[BT7] 162:     while (z[a] = SPACE) or (z[a] = TAB) do a
:= a + 1;
[2Z3] 163:     for i := a to length(z) do
[AP2] 164:         if z[i] < #32 then
```

```
[0G2] 165:         for j := 1 to 3 do
[MU0] 166:         begin
[TZ1] 167:             GotoXY(3,21);
[EC9] 168:             write(' +++++ WARNUNG: Steuerzeic
hen in Zeile !
', BEL);
[HN1] 169:             delay(200);
[TZ1] 170:             GotoXY(3,21);
[FS4] 171:             write('
');
[HN1] 172:             delay(200);
[4M0] 173:         end
[AZ7] 174:         else if eins then p := ((i-a+1) and 7)
* ord(z[i]) + p
else p := (((i-a+1) and 3) + 1) *
[PN7] 175:         ord(Upper(z[i])) + p;
[8H1] 176:         ps := ' ';
[L12] 177:         ps[1] := '[';
[5G2] 178:         for i := 2 to 4 do
[MU0] 179:         begin
[475] 180:             ps[i] := ZEICHEN[(p and 31) + 1];
[TP1] 181:             p := p shr 5;
[ZN0] 182:         end;
[Z12] 183:         ps[5] := ']';
[ZN0] 184:     end;
[000] 185:
[000] 186:
[MG5] 187: procedure edit(var z: ZeilenTyp);
[000] 188:
[012] 189: const s = #32;
[000] 190:
[RN0] 191: var
[6W1] 192:     x: integer;
[GC1] 193:     ch: char;
[000] 194:
[2H4] 195: procedure schreibe(c: char);
[MU0] 196: begin
[835] 197:     if length(z) = 255 then write(BEL)
[ZR0] 198:     else
[MU0] 199:     begin
[3G2] 200:         insert(c, z, x);
[7F1] 201:         x := x + 1;
[GV1] 202:         GotoXY(1,1);
[5H1] 203:         write(z);
[ZN0] 204:     end;
[ZN0] 205: end;
[000] 206:
[Z22] 207: procedure rechts;
[MU0] 208: begin
[025] 209:     if x > length(z) then write(BEL)
[G82] 210:     else x := x + 1;
[ZN0] 211: end;
[000] 212:
[SV2] 213: procedure links;
[MU0] 214: begin
[5N3] 215:     if x = 1 then write(BEL)
[M82] 216:     else x := x - 1;
[ZN0] 217: end;
[000] 218:
[MN2] 219: procedure hoch;
[MU0] 220: begin
[PL3] 221:     if x < 78 then write(BEL)
[YC2] 222:     else x := x - 77;
[ZN0] 223: end;
[000] 224:
[C03] 225: procedure runter;
[MU0] 226: begin
[BB5] 227:     if x + 77 > length(z) then write(BEL)
[UC2] 228:     else x := x + 77;
[ZN0] 229: end;
[000] 230:
[3P2] 231: procedure entf;
[MU0] 232: begin
[2B3] 233:     if x <= length(z) then
[MU0] 234:     begin
[L92] 235:         delete(z, x, 1);
[GV1] 236:         GotoXY(1,1);
[ZX1] 237:         write(z,s);
[4M0] 238:     end
[VL2] 239:     else write(BEL);
[ZN0] 240: end;
[000] 241:
[TC2] 242: procedure del;
[MU0] 243: begin
[5N3] 244:     if x = 1 then write(BEL)
[ZR0] 245:     else
[MU0] 246:     begin
[WG2] 247:         delete(z, x-1, 1);
[9F1] 248:         x := x - 1;
[GV1] 249:         GotoXY(1,1);
[ZX1] 250:         write(z,s);
[ZN0] 251:     end;
[ZN0] 252: end;
[000] 253:
[FN2] 254: procedure pos1;
```

Listing »scheck.pas« in Turbo Pascal 3.0

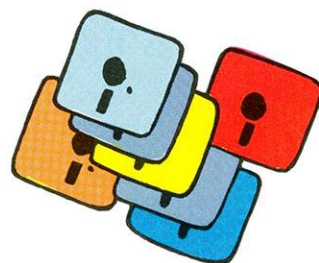


Hardware

Software

Dazu gehört

InfoWare



InfoWare – Die ideale **Ergänzung zu Ihren Anwendungsprogrammen!** Gut informiert zu sein, bedeutet schnelle und erfolgreiche Arbeit, erspart viel Suchen und gewährt den entscheidenden Vorsprung.

InfoWare ist ein leicht zu installieren-**des Online-Hilfssystem** für eine große Anzahl von am Markt erhältlichen Softwareprodukten namhafter Hersteller.

Sie werden begeistert sein und InfoWare schon bald nicht mehr missen wollen!

InfoWare wartet als **speicherresidentes Programm** unsichtbar im Hintergrund, bis Sie seine Hilfe benötigen. Dann ist es allerdings **blitzschnell** per Tastendruck **zur Stelle**, um Ihnen bei jedem Problem zu helfen. Die ausgefeilte **Fenstertechnik** erlaubt über

Stichworte oder Querverweise eine optimale Bedienerhilfe.

Neben den in InfoWare bereits vorhandenen Einträgen können bis zu 50 weitere umfangreiche Texthilfen selbst integriert werden.

Deshalb ist InfoWare das **unverzichtbare Hilfsmittel** für alle Programm-anwender!

InfoWare-Hilfsfenster
Klarheit per Tastendruck

Ausführliche Informationen zum Suchbegriff, max. 1 Bildschirmseite und eigene Einträge

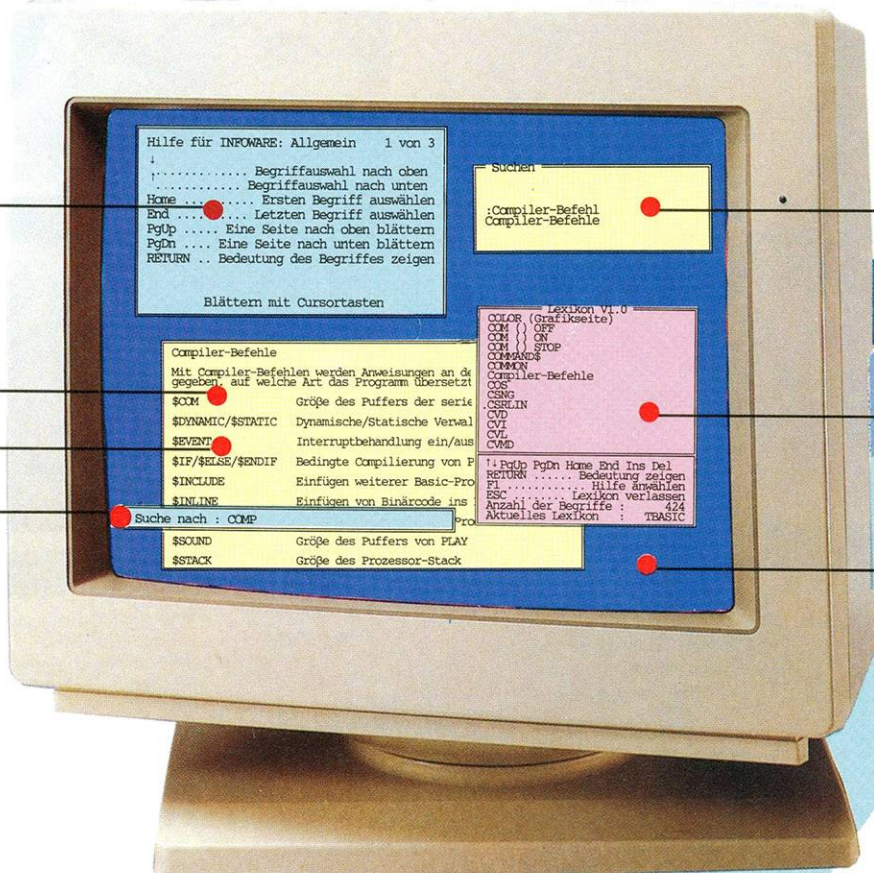
Querverweise

Blitzschneller Aufruf des gesuchten Begriffes

Suchraster erleichtert das Auffinden von Einträgen

Pull-Down-Menü zur Auswahl des Suchbegriffes. Bedienung denkbar einfach über Cursor-Tasten

Speicherresident



Folgende Titel erscheinen in Kürze

dBase III Plus *3	Star Writer *7
Clipper *6	Word-Workshop *1
Fox BASE *5	WordPerfect-Workshop *4
Multiplan *1	Open Access II *8
Turbo Pascal 4.0 *2	Framework II *3
Modula II	

Hinweis: Die genannten Produkte sind eingetragene Warenzeichen von:

*1 – Microsoft, *2 – Borland, *3 – Ashton Tate,
*4 – Wordperfect-Corporation, *5 – Fox Software Inc.,
*6 – Nantucket, *7 – Star Division, *8 – SPI

Notwendige Konfiguration:

IBM-kompatiblen PC XT/AT mit 512 kByte oder mehr, 1 Diskettenlaufwerk bzw. Festplatte, PC-DOS/MS-DOS ab Version 2.0, mit Monochrom- oder Farbbildschirm

* Unabhängig von der Anzahl der bestellten Produkte berechnen wir für das Inland 3,- DM bzw. für das Ausland 5,- DM Porto und Verpackung.
Bitte benutzen Sie die Bestellkarte.

© 1988 DMV, Eschwege
Licensed by infoPRO, infoPRO ist ein Warenzeichen der HOMESOFT GmbH & Co.KG, München



unverbindliche Preisempfehlung

InfoWare ist für nur **69,-* DM** pro Titel für folgende Programme erhältlich:

MS-DOS *1	Lotus 1-2-3 *3
PC-DOS *1	MS-Word
GW Basic *1	3.0/4.0-Grundlagen *1
Quick Basic *1	WordPerfect
Turbo Basic *2	4.2-Grundlagen *4
Quick C *1	PC-Grundlagen
Turbo C *2	C-Grundlagen
Assembler 8086/88	

erhältlich auf MS-DOS 5 1/4"- oder 3 1/2"-Diskette

DMV GmbH, Fuldaer Str. 6, Postfach 250, 3440 Eschwege


```
[MU0] 255: begin
[M20] 256:   x := 1;
[ZN0] 257: end;
[000] 258:
[0N2] 259: procedure ende;
[MU0] 260: begin
[6T2] 261:   x := length(z) + 1;
[ZN0] 262: end;
[000] 263:
[MU0] 264: begin
[TZ1] 265:   GotoXY(3,21);
[LLB] 266:   write(' Editieren Sie die Programmzeile
                                [RETURN] Ende ')
;
[6K2] 267:   window(2,6,78,10);
[DB1] 268:   ClrScr;
[5H1] 269:   write(z);
[GV1] 270:   GotoXY(1,1);
[M20] 271:   x := 1;
[P31] 272:   repeat
[722] 273:     read(kbd, ch);
[LM1] 274:     case ch of
[9P0] 275:       ESC:
[MU0] 276:         begin
[722] 277:           read(kbd, ch);
[LM1] 278:           case ch of
[BX1] 279:             'M': rechts;
[PS1] 280:             'K': links;
[ME1] 281:             'H': hoch;
[6Z1] 282:             'P': runter;
[BE1] 283:             'G': pos1;
[RE1] 284:             'O': ende;
[NF1] 285:             'S': entf;
[ZN0] 286:           end;
[ZN0] 287:         end;
[7A1] 288:         BS: del;
[UR0] 289:         CR: ;
[YX2] 290:         else schreibe(ch);
[ZN0] 291:       end;
[VG4] 292:       GotoXY(x mod 77, x div 77 + 1);
[842] 293:       until ch = CR;
[XZ6] 294:       if auto then z := copy(check, 2, 3) + SPA
CE + z;
[ZN0] 295: end;
[000] 296:
[000] 297:
[DP3] 298: procedure ZeigeZeile;
[MU0] 299: begin
[902] 300:   if auto then
[MU0] 301:     begin
[SC5] 302:       check := '[' + copy(Zeile, 1, 3) + ']'
;
[RP4] 303:       check[2] := UpCase(check[2]);
[YP4] 304:       check[3] := UpCase(check[3]);
[5R4] 305:       check[4] := UpCase(check[4]);
[9A4] 306:       Zeile := copy(Zeile, 5, 255);
[ZN0] 307:     end;
[P42] 308:     GotoXY(29,12);
[NC2] 309:     write(Nummer);
[6K2] 310:     window(2,6,78,10);
[DB1] 311:     ClrScr;
[M72] 312:     write(Zeile);
[GH2] 313:     window(1,1,80,25);
[LB2] 314:     pruefe(Zeile);
[342] 315:     GotoXY(10,12);
[MN1] 316:     write(ps);
[DN2] 317:     if bin then exit;
[902] 318:     if auto then
[5S2] 319:       if ps = check then
[MU0] 320:         begin
[TZ1] 321:           GotoXY(3,21);
[9B7] 322:           write(' Prüfsumme OK für Zeile ',
Nummer);
[GG2] 323:           weiter := true;
[4M0] 324:         end
[ZR0] 325:         else
[MU0] 326:           begin
[TZ1] 327:             GotoXY(3,21);
[RBE] 328:             write(BEL, ' Prüfsummenfehler !
[E] Editieren [RETURN] Weiter [Ctrl-C]
Abbrechen');
[D42] 329:             GotoXY(15,12);
[4M0] 330:           end
[ZR0] 331:           else
[MU0] 332:             begin
[TZ1] 333:               GotoXY(3,21);
[5ZC] 334:               write(' [E] Editieren [RETURN] W
eiter [Q] Quitt [Ctrl-C] Abbreche
n');
[D42] 335:               GotoXY(15,12);
[ZN0] 336:             end;
[ZN0] 337:           end;
[000] 338:
[000] 339:
[Y23] 340: procedure Beenden;
[S42] 341: var fi: file;
```

```
[MU0] 342: begin
[R43] 343:   while not eof(t) do
[MU0] 344:     begin
[ZN2] 345:       readln(t, Zeile);
[LL3] 346:       writeln(temp, Zeile);
[ZN0] 347:     end;
[LL3] 348:     close(t); close(temp);
[Z63] 349:     if bin then close(f);
[CS3] 350:     assign(fi, Name + '.BAK');
[4T3] 351:     {$I-} reset(fi); {$I+}
[EX4] 352:     if IOresult = 0 then erase(fi);
[VE3] 353:     rename(t, Name + '.BAK');
[SY3] 354:     rename(temp, Name + Ext);
[TZ1] 355:     GotoXY(3,21);
[MW3] 356:     if bin then Ext := '.BIN';
[FH8] 357:     write(' Die Datei ', Name, Ext, ' wurde g
espeichert ');
[KR4] 358:     while WhereX < 78 do write(' ');
[702] 359:     GotoXY(1,25);
[DU0] 360:     halt;
[ZN0] 361:   end;
[000] 362:
[000] 363:
[DX2] 364: procedure Quitt;
[MU0] 365: begin
[TZ1] 366:   GotoXY(3,21);
[SBB] 367:   write(' Drücken Sie [J], um die Arbeit zu
beenden ');
[T42] 368:   GotoXY(48,21);
[S02] 369:   read(kbd, ch);
[NL1] 370:   write(ch);
[464] 371:   if (ch = 'j') or (ch = 'J') then
[MU0] 372:     begin
[LL3] 373:       writeln(temp, Zeile);
[ZB1] 374:       Beenden;
[ZN0] 375:     end;
[ZN0] 376:   end;
[000] 377:
[000] 378:
[MU0] 379: begin
[AV0] 380:   init;
[DS1] 381:   if bin then
[MU0] 382:     begin
[TZ1] 383:       GotoXY(3,21);
[N97] 384:       write(' >>> Binärdatei wird erzeugt <
<< ');
[ZN0] 385:     end;
[R43] 386:   while not eof(t) do
[MU0] 387:     begin
[ZN2] 388:       readln(t, Zeile);
[P31] 389:       repeat
[2K2] 390:         weiter := false;
[P12] 391:         ZeigeZeile;
[X94] 392:         if bin then weiter := true;
[DX2] 393:         if not weiter then
[MU0] 394:           begin
[722] 395:             read(kbd, ch);
[LM1] 396:             case ch of
[853] 397:               CR: weiter := true;
[P13] 398:               'e','E': Edit(Zeile);
[V65] 399:               'q','Q': if not auto then Qui
tt;
[ZN0] 400:             end;
[ZN0] 401:           end;
[G92] 402:           until weiter;
[LL3] 403:           writeln(temp, Zeile);
[DS1] 404:           if bin then
[MU0] 405:             begin
[E68] 406:               while copy(Zeile, 1, 1) = SPACE d
o delete(Zeile, 1, 1);
[P31] 407:               repeat
[W55] 408:                 val('$' + copy(Zeile, 1, 2), b,
code);
[5L5] 409:                 if (code > 0) or (b < 0) or (b
> 255) then
[MU0] 410:                   begin
[TZ1] 411:                     GotoXY(3,21);
[ZV9] 412:                     write(' +++++ ABBRUCH WEGEN
DATENFEHLER +++++
');
[702] 413:                     GotoXY(1,25);
[DU0] 414:                     halt;
[ZN0] 415:                   end;
[631] 416:                   by := b;
[B32] 417:                   write(f, by);
[3A4] 418:                   Zeile := copy(Zeile, 3, 255);
[E68] 419:                   while copy(Zeile, 1, 1) = SPACE
do delete(Zeile, 1, 1);
[KF3] 420:                   until Zeile = EMPTY;
[ZN0] 421:                 end;
[C43] 422:                 Nummer := Nummer + 1;
[ZN0] 423:               end;
[ZB1] 424:             Beenden;
[KN0] 425:           end.
```

Listing »scheck.pas« in Turbo Pascal 3.0

Für jeden Computerfreund etwas dabei



Jetzt neu
PCpur für Einsteiger
und Aufsteiger

DOS International

Die erfolgreiche und universale Zeitschrift für Anwender und Programmierer aller PCs unter MS-DOS.

WINDOW

Das neue Magazin, das Maßstäbe für den Umgang mit anspruchsvoller Software setzt. Von Profis gemacht, für Anwender gedacht!

PC Amstrad International

Die Nummer Eins, wenn es um CPCs, PCW Joyce und PC 1512/1640 geht. Im 4. Jahrgang und beliebt wie eh und je.

Die Sonderhefte

vertiefen die Themen, die oft zu knapp behandelt werden und liefern die ideale Grundlage für besondere Anwendungen.

TOOLBOX

Keine andere Zeitschrift setzt sich so konsequent für moderne Computersprachen ein! Programmierer haben hier ihr Medium.

JOYSTICK

Spiele ohne JOYSTICK sind wie Pacman ohne Geister! Das Magazin für Computerunterhaltung und alles, was Computern Spaß macht.

PC pur

Sie brauchen kein EDV-Profi zu sein, um den PC zu verstehen. PC pur zeigt Einsteigern und Umsteigern den Weg.

DOS Extra

CPC Sonderhefte

JOYCE Sonderhefte

DMV Verlag · Postfach 250 · 3440 Eschwege

GW-Basic öffnet die Fenster

Im Befehlsumfang von GW-Basic sind keine Fenster-Routinen und Menübefehle enthalten, die eine bedienungsfreundliche Benutzeroberfläche ermöglichen. Mit einer Hilfsroutine, die Interrupt-Befehle simuliert, rufen Sie die Betriebssystemroutinen des PC auf und können alle Vorzüge des Betriebssystems genießen. Die Mankos des Basic-Interpreters werden damit behoben.

Das BIOS (Basic Input Output System) im PC und das Betriebssystem MS-DOS stellen dem Programmierer leistungsfähige Routinen für Tastatur-, Bildschirm-, Datei-, Datenträger- und Schnittstellenverwaltung zur Verfügung. Doch diese Routinen lassen sich nur in Assemblerprogrammen und einigen Hochsprachen direkt aufrufen; dem GW-Basic-Programmierer bleiben sie normalerweise verschlossen.

Fast alle BIOS-Routinen sind Interruptroutinen oder Funktionen von Interruptroutinen – das heißt, sie lassen sich über Unterbrechungsanforderungen der Peripherie oder über die sogenannten Software-Interrupts 00 bis FFhex aufrufen. Das Arbeiten mit Hardware-Interrupts erhöht die Leistungsfähigkeit des Computers, denn die Peripherie nimmt die CPU nur dann in Anspruch, wenn es wirklich notwendig ist. Ebenso bietet das Konzept der Software-Interrupts wesentliche Vorteile. Sind die Interruptroutinen mit den Routinen des Betriebssystems belegt, lassen sich die Betriebssystemfunktionen über einfache Interruptbefehle aufrufen.

Der GW-Basic-Programmierer hat hier allerdings das Nachsehen, weil unter GW-Basic keine Befehle für Interruptaufrufe existieren. Fenster-Scrolling, die Abfrage der Systemkonfiguration oder Manipulationen am Betriebssystem sind somit nahezu unmöglich. Mit dem Maschinensprachenprogramm »interrupt« können Sie aber die Interruptroutinen und damit die Betriebssystemroutinen doch nutzen. Diese neugewonnene Fähigkeit von GW-Basic eröffnet Ihnen Anwendungsbereiche, die weit über das üblichen Einsatzgebiet des Basic-Interpreters hinausgehen.

Listing 1 zeigt »interrupt« in der Assembler-Fassung. Sie müssen jedoch keinen Assembler besitzen, um »interbas« zu benutzen. In Listing 2 ist »interrupt« in ein Basic-Programm eingebunden. In diesem Fall liegt die Hilfsroutine in den Data-Zeilen als Maschinencode vor. GW-Basic schreibt diesen Maschinencode beim Programmstart in den Arbeitsspeicher des PC und hält die Routine dort für den Aufruf durch einen Call-Befehl bereit. Sie müssen also die Basic-

Zeilen von »interrupt« lediglich in das Programm einbinden, aus dem Sie die Interruptroutinen des PC aufrufen wollen. Die Zeilennummern dürfen Sie Ihren eigenen Wünschen anpassen. Listing 3 bedient sich des Unterprogramms, um den Systemstatus zu ermitteln. Wie das aussieht, sehen Sie in dem Bild.

Grundsätzlich muß »interbas« durch einen eigenen Speicherbereich (Anwenderstapel) unterstützt werden. Dieser Speicherbereich wird in dem Programm durch die Variable »stapel\$« repräsentiert. Die dazu einzufügenden Zeilen lauten beispielsweise:

```
100 Nummer.des.Interrupts=&H10
110 stapel$ = SPACE$(128)
```

Übergeben wird die Nummer des Interrupts, die Register und der Stapelspeicher

Sie geben in dem Beispiel als Interruptnummer den Wert 10hex an und reservieren 128 Byte für den Anwenderstapel. Diese Funktion scrollt einen Bildschirmbereich. Die Stapelgröße von 128 Byte reicht selbst für aufwendige Anwendungen aus. Als nächstes müssen Sie die relevanten Register mit den Übergabeparametern belegen. Die Werte der übrigen Register sind gleichgültig. Der Video-Interrupt 10hex, Funktion 6, benötigt Parameter in den Registern

AH, AL, BH, CH, CL, DH und DL. AH enthält die Funktionsnummer, also den Wert 6. AL gibt die Anzahl der Zeilen an, die gescrollt werden sollen. In CH und CL stehen die Koordinaten der linken oberen sowie in DH und DL die Koordinaten der rechten unteren Ecke. Schließlich müssen Sie in BH noch das Attribut für die gelöschten Zeilen angeben. Als letzten Parameter führen Sie einen Rückgabeparameter an, der jedoch nur wichtig ist, wenn die aufgerufene Routine einen Fehler zurückgibt.

Folgende Werte scrollen das Fenster zwischen den Koordinaten 2,2 und 20,20 um fünf Zeilen aufwärts und weisen den dadurch freiwerdenden Zeilen das Attribut Null zu:

```
120 AH=6
130 AL = 5
140 CH = 2
150 CL = 2
160 DH = 20
170 DL = 20
180 BH = 0
```

Wenn Sie als nächstes

```
190 rueckgabeparameter=255
```

in das Programm einfügen, dann ist nach dem Aufruf der Routine immer dann kein Fehler aufgetreten, wenn die Variable »rueckgabeparameter« den Wert 255 behält. Hat sich der Wert verändert, so ist während des Ablaufs der Routine ein Fehler aufgetreten.

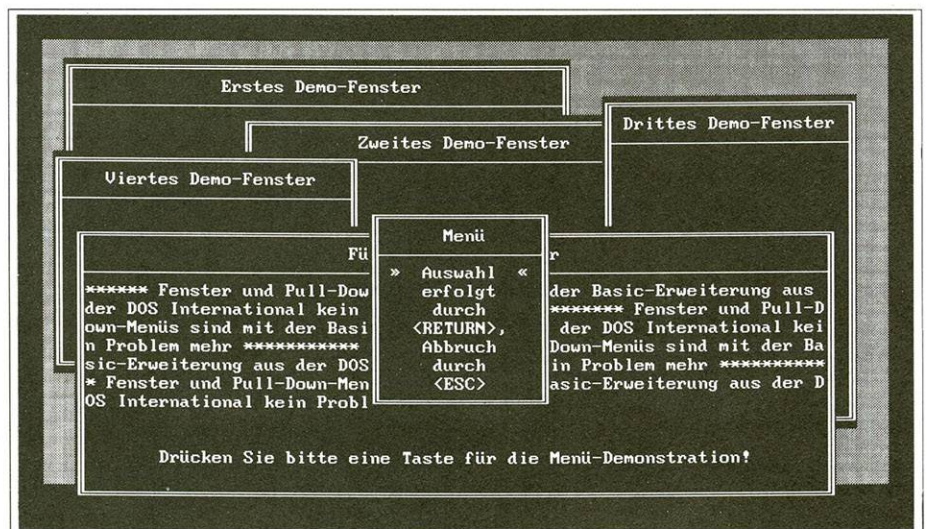
In der Variablen »Nummer.des.Interrupts« legen Sie die Nummer des aufzurufenden Interrupts ab. Haben Sie das Hauptprogramm mit den aufgeführten Variablenzuweisungen versorgt, so erfolgt der Aufruf der Betriebssystemroutine durch

```
200 GOSUB 10000
```

Vergessen Sie danach nicht, Ihr Programm ordnungsgemäß mit

```
210 END
```

abzuschließen.



Fenstertechnik geht auch mit GW-Basic

Ab Zeile 10000 müssen Sie dann die entsprechenden Anweisungen aus dem Programm »interbas« einfügen. Das Programm ist damit fertig.

Der Programmiersprache GW-Basic fehlen einige moderne Leistungsmerkmale wie Pull-Down-Menüs oder Fenstertechnik. Die Basic-Erweiterung »erweiter« stellt Ihnen Routinen dafür zur Verfügung. Mit den Prozeduren können Sie eigene Benutzeroberflächen entwickeln.

Dazu ein Beispiel:

Sie wollen in einem Ihrer Programme Hilfe-Fenster auf Tastendruck bereitstellen. Um den Informationsbereich auf dem Bildschirm darzustellen, geben Anwendungsprogramme in den meisten Fällen einen Kasten aus, der die gewünschten Angaben enthält. Nach der Rückkehr in den Editor muß der vorherige Inhalt des Bildschirms wiederhergestellt werden. Ähnlich funktioniert die Arbeit mit einem Pull-Down-Menü. Nachdem es »heruntergeklappt« ist und der Benutzer einen Menüpunkt angewählt hat, verschwindet es wieder, und der alte Bildschirminhalt muß wieder zu sehen sein.

Hilfefunktionen in Bildschirmfenstern sind ein Leistungsmerkmal anwenderfreundlicher Programme

Die Routinen aus Listing 3 stellen Ihnen Unterprogramme zur Verfügung, die es erlauben, eine ähnliche Menütechnik in eigenen Programmen zu verwenden. Die Prozeduren verwenden aussagefähige Variablen und geben Ihnen so einen Einblick in die Leistungsfähigkeit des Routinenpakets.

Wenn Sie das Programm starten, initialisiert die Routine die benötigten Variablen und richtet einen Zwi-

schenspeicher namens »ffld« ein. In »ffld« legt das Programm den Inhalt der einzelnen Fenster ab, damit diese nach dem Schließen eines anderen Fensters wieder restauriert werden können. Die Konstante »fgrenze« bestimmt die maximale Größe des Fensterstapels. Im vorliegenden Beispiel ist er auf 4000 ASCII-Zeichen begrenzt. Reicht dieser Platz nicht aus, dann ändern Sie einfach den Grenzwert im Definitionsteil des Listings. Die wichtigste Routine ist die Prozedur ab Zeile 50000, die ein neues Fenster öffnet und den Inhalt des darunterliegenden Bildschirmbereichs sichert. Als Parameter übergeben Sie die linke, rechte, obere und untere Grenze des Fensters mit Hilfe der entsprechenden Variablen. Das letzte Argument ist die Titelzeile des Fensters in der Variablen »text\$«. Wenn die Länge des Titels die Breite des Fensters übersteigt, dann kürzt die Prozedur den Namen automatisch, anderenfalls bildet sie ihn zentriert ab.

Bevor ein Fenster geöffnet wird, stellt es durch den Aufruf von »status« in der Variablen »zzeile« die Anzahl der Zeichen pro Zeile fest. Das Unterprogramm, das ein Fenster schließt, stellt den vorherigen Bildschirminhalt wieder her. Die Routine übernimmt keine Parameter. Wenn nach dem Schließen des Fensters noch ein weiteres geöffnet ist, dann wird der Cursor auf die Position gesetzt, die zuletzt in dem Fenster festgelegt wurde.

Mit der nächsten Routine bestimmen Sie die Position des Cursors innerhalb eines Fensters. Wie bei dem Locate-Befehl von GW-Basic übergeben Sie die y- und die x-Koordinate der Eingabemarke in den zugehörigen Variablen. Die Angabe bezieht sich immer relativ auf die linke obere Ecke des Fensters, die die Koordinaten 1,1 erhält. Sie können die aktuellen Koordinaten des Cursors abfra-

gen, indem Sie den Inhalt der Variablen »f.y« und »f.x« betrachten. Die Prozedur, die das aktuelle Fenster löscht, benötigt keine Argumente. Sie setzt zusätzlich den Cursor auf die linke, obere Ecke des Fensters. Das sind die relativen Koordinaten 1,1. Die Ausgabe-Routine gibt den in der Variablen »text\$« übergebenen Text an der Cursor-Position

aus. Sie erhöht die x- und y-Koordinate um den der Länge des Textes entsprechenden Wert. Erreicht der Cursor die letzte Fensterzeile, scrollt er den Bildschirm mit dem dafür vorgesehenen Unterprogramm um eine Zeile aufwärts und setzt die Ausgabe fort.

Sie können die Scroll-Routine auch gesondert aufrufen. Zu diesem Zweck legen Sie in der Variablen »farbe%« das Farbattribut ab, das die Prozedur in die letzte Zeile des verschobenen Bereichs schreibt. Die Programmzeilen beziehen sich immer auf das aktuelle Fenster.

Fenster-Technik in GW-Basic-Programmen professionalisieren Ihr Anwendungsprogramm

Zum Schluß stellt die Prozedur-Sammlung für Pull-Down-Menüs ein Menüunterprogramm zur Verfügung. Sie benötigt keine Parameter und versetzt den Anwender in die Lage, das aktuelle Fenster als Pull-Down-Menü zu benutzen. Er wählt mit den Cursor-Tasten den gewünschten Menüpunkt aus und bestätigt ihn durch das Drücken des Enter-Taste. Nach dem Aufruf der Routine wird die Nummer der ausgewählten Menüzeile in der Variablen »m.punkt« abgelegt. Wenn der Anwender die Taste »Esc« drückt, dann erhält diese Variable den Wert Null.

Wenn Sie die Reaktion auf falsche Parameterübergaben verändern wollen, müssen Sie das Unterprogramm, das die Fehler behandelt, entsprechend ändern. Es ist beispielsweise denkbar, einen Programmteil zu konzipieren, der, ähnlich dem GW-Basic-Kommando »on error goto«, jede Fehlermeldung abfängt.

Die Bildschirm-Routinen des Fensterpakets unterstützen alle wichtigen Grafikkarten, da sie im Text-Modus arbeiten. Den verschiedenen Bildschirmbreiten ist mit der Entwicklung der Fenster- und Menüroutinen Rechnung getragen. Wenn Sie beispielsweise mit dem Screen-Kommando vierzig Zeichen pro Zeile einstellen, dann berücksichtigen die Prozeduren die geänderten Werte und passen die Unterprogramme an das neue Bildschirmformat an. Mit dem Unterprogrammpaket steht Ihnen somit ein leistungsfähiges Hilfsmittel zur Verfügung, das die Programmierung von bedienungsfreundlichen Anwendungen erleichtert.

(Markus Zietlow/db)

```
[415] 1: ; Assemblerprogramm für den
[235] 2: ; Interrupt-Aufruf unter Basic
[P30] 3: ;
[WP3] 4: ; Aufrufform in Basic:
[92E] 5: ; CALL INTERRUPT(AH,AL,BH,BL,CH,CL,DH,DL,In
      ; terruptnummer,Rückgabeparameter,Stapel$)

[P30] 6: ;
[P30] 7: ;
[P30] 8: ;
[0L1] 9: ; page 72,132
[3E5] 10: cseg segment para public 'code'
[GR2] 11: public intrrpt
[ZW2] 12: intrrpt proc far
[A88] 13: assume cs:cseg,ds:nothing,ss:nothing,
      ; es:nothing
[518] 14: push bp ;-Basiszeiger sic
      ; hern und auf die
[E07] 15: mov bp,sp ; Übergabeparamet
      ; er richten
[ZT6] 16: mov di,sp ;-Basic-Stapel üb
      ; ernehmen
[MR3] 17: mov si,[bp+6] ;-Neuen
[NY3] 18: mov al,[si] ; Stapel
[LT3] 19: mov ah,0 ; einlesen
[VU3] 20: inc si ; und für
```



```

[3D3] 21:      add ax,[si]      ; den
[C65] 22:      dec ax          ; Interrupt-Aufru
f
[HP4] 23:      mov sp,ax       ; vorbereiten
[CX5] 24:      push di         ; -Basic-Stapel si
chern
[D26] 25:      mov ax,0        ; -Den Befehlszeig
er, der
[2C5] 26:      mov ds,ax       ; auf die Origina
l-
[SG7] 27:      mov ax,[ds:144] ; Fehlerbehandlun
gsroutine
[F85] 28:      push ax         ; zeigt, retten u
nd
[7S3] 29:      mov ax,[ds:146] ; den
[GK3] 30:      push ax         ; Zeiger
[854] 31:      call routine    ; auf die
[CB7] 32: routine:pop bx      ; Fehlerabfangrou
tine
[7N3] 33:      db 129,195,86,0 ; setzen,
[L44] 34:      mov ds:90h,bx   ; die sich
[X3] 35:      push cs          ; in diesem
[4U3] 36:      pop ax          ; Programm
[E54] 37:      mov ds:92h,ax   ; befindet
[2R4] 38:      mov ax,es       ; -Datensegment
[3F5] 39:      mov ds,ax       ; wiederherstelle
n
[4U6] 40:      mov si,[bp+26]  ; -Die einzelnen R
egister
[T17] 41:      mov ah,[si]     ; von ah bis dl ü
bernehmen
[WA2] 42:      mov si,[bp+24]
[532] 43:      mov al,[si]
[SA2] 44:      mov si,[bp+22]
[V22] 45:      mov bh,[si]
[MA2] 46:      mov si,[bp+20]
[732] 47:      mov bl,[si]
[3B2] 48:      mov si,[bp+18]
[X22] 49:      mov ch,[si]
[ZA2] 50:      mov si,[bp+16]
[932] 51:      mov cl,[si]
[VA2] 52:      mov si,[bp+14]
[Z22] 53:      mov dh,[si]
[RA2] 54:      mov si,[bp+12]
[B32] 55:      mov dl,[si]
[485] 56:      push ax         ; -Register ax und b
x
[DS6] 57:      push bx         ; vor Änderungen sc
hützen
[EM6] 58:      mov si,[bp+10]  ; -Interruptnummer i
n das
[L26] 59:      mov al,[si]     ; al-Register einla
den
[XV8] 60:      call routin2    ; -Aktuellen Befehls
zeiger feststellen
[X77] 61:      routin2:pop bx   ; und in bx abspeic
hern
[HV7] 62:      db 129,195,14,0 ; -Setzt Zeiger auf
Interruptaufruf
[FA7] 63:      mov cs:[bx],al  ; und speichert die
in Basic
[N68] 64:      jmp short inter  ; angegebene Interr
uptnummer ab
[EP6] 65:      inter: pop bx    ; -Angegebene Regist
er
[F15] 66:      pop ax          ; wiederherstellen
[PX8] 67:      push bp          ; -Adresse des Rückg
abparameters sichern
[DV5] 68:      db 205,0        ; -Interrupt aufrufe
n und
[3CA] 69:      jmp weiter      ; weitermachen, wen
n kein Fehler aufgetreten ist
[MY4] 70:      sti             ; -Fehlerbehandlung:
[BH6] 71:      mov ax,di        ; -Nummer des aufget
retenen
[6F6] 72:      mov ah,0        ; Fehlers in AL abs
peichern
[0Z7] 73:      add sp,18       ; -Unbenutzten Stape
l überspringen
[463] 74:      pop bp          ; und die
[KR5] 75:      pop ds          ; benötigten Regist
er
[815] 76:      pop es          ; wiederherstellen
[DE7] 77:      add sp,6         ; -Stapel weiter ver
kleinern und
[V0C] 78:      pop bp          ; den Zeiger auf di
e übergebenen Parameter wiederherstellen
[TC4] 79:      mov si,[bp+8]   ; -Fehlercode
[NG8] 80:      mov [si],ax     ; im Rückgabeparamete
r abspeichern
[MS9] 81:      jmp abbruch     ; -Aktualisieren der
Register überspringen
[189] 82:      weiter: pop bp   ; -Parameterzeiger w
iederherstellen
[AK6] 83:      mov si,[bp+26]  ; -Die einzelnen Reg
ister

```

```

[8U4] 84:      mov [si],ah     ; von ah bis dl
[W36] 85:      mov si,[bp+24]  ; in den zugehörige
n
[C85] 86:      mov [si],al     ; Basic-Variablen
[7L4] 87:      mov si,[bp+22]  ; abspeichern
[D32] 88:      mov [si],bh
[MA2] 89:      mov si,[bp+20]
[X32] 90:      mov [si],bl
[3B2] 91:      mov si,[bp+18]
[G32] 92:      mov [si],ch
[ZA2] 93:      mov si,[bp+16]
[042] 94:      mov [si],cl
[VA2] 95:      mov si,[bp+14]
[L32] 96:      mov [si],dh
[RA2] 97:      mov si,[bp+12]
[342] 98:      mov [si],dl
[SH6] 99:      abbruch:mov ax,0 ; -Den Befehlszeiger,
[W54] 100:     mov ds,ax       ; der auf die
[4U6] 101:     pop ax          ; Original-Fehlerbeha
ndlungs-
[FH6] 102:     mov ds:92h,ax   ; routine zeigt, sowi
e das
[1M4] 103:     pop ax          ; ursprüngliche
[146] 104:     mov ds:90h,ax   ; Code-Segment erneue
rn
[NU6] 105:     mov ax,es       ; -Das vorherige Daten
segment
[355] 106:     mov ds,ax       ; wiederherstellen
[6F5] 107:     pop di         ; -Den Original-Stapel
[UG3] 108:     mov sp,di      ; und den
[9D6] 109:     pop bp         ; Basiszeiger zurückh
olen
[EB6] 110:     ret 16H        ; -Zurück ins Basic-Pr
ogramm
[000] 111:
[P72] 112: intrrpt endp
[HR1] 113: cseg ends
[4M0] 114: end

```

Listing 1. Das Assembler-Listing »interrupt.asm« zeigt Ihnen den Aufbau der Interrupt-Routine

```

[ML9] 1 '*** Programm für den Interruptaufruf unter GW-
Basic ***
[AXA] 2 'Bevor ein Interrupt aufgerufen werden kann, mü
ssen zuerst alle
[HMA] 3 'benötigten Register initialisiert werden (AH,A
L,BH,BL,CH,CL,
[M69] 4 'DH,DL). Deswegen müssen Sie einen neuen Stapel
für das
[PVA] 5 'Maschinenspracheunterprogramm und die Interru
ptnummer sowie
[LA5] 6 'den Rückgabeparameter angeben.
[AB0] 7 '
[H03] 8 'Initialisierung
[EB0] 9 '
[HV1] 10 DEFINT A-Z
[VK1] 20 RESTORE
[R9D] 30 READ CODE$:IF CODE$<>"ENDE" THEN INTERRUPT$=IN
TERRUPT$+CHR$(VAL("&h"+CODE$)):GOTO 30
[AA1] 40 DEF SEG
[0H0] 100 '
[Y22] 200 'Hauptprogramm
[4H0] 300 '
[W21] 400 END
[8H0] 500 '
[1N7] 10000 'Unterprogramm für den Interruptaufruf
[ZT0] 10010 '
[C2C] 10020 INTERRUPT!=PEEK(VARPTR(INTERRUPT$)+2)*256+P
EEK(VARPTR(INTERRUPT$)+1)
[KFF] 10030 CALL INTERRUPT!(AH,AL,BH,BL,CH,CL,DH,DL,NUM
MER.DES.INTERRUPTS,RUECKGABEPARAMETER,STAPEL$)
[GV1] 10040 RETURN
[8U0] 60000 '
[NH7] 60010 'Daten für das Maschinenspracheprogramm
[AU0] 60020 '
[C65] 60030 DATA 55,8b,ec,8b,fc,8b,76,06,8a,04
[M15] 60040 DATA b4,00,46,03,04,48,8b,e0,57,b8
[705] 60050 DATA 00,00,8e,d8,a1,90,00,50,a1,92
[RY4] 60060 DATA 00,50,e8,00,00,5b,81,c3,56,00
[105] 60070 DATA 89,1e,90,00,0e,58,a3,92,00,8c
[G55] 60080 DATA c0,8e,d8,8b,76,1a,8a,24,8b,76
[345] 60090 DATA 18,8a,04,8b,76,16,8a,3c,8b,76
[F55] 60100 DATA 14,8a,1c,8b,76,12,8a,2c,8b,76
[945] 60110 DATA 10,8a,0c,8b,76,0e,8a,34,8b,76
[115] 60120 DATA 0c,8a,14,50,53,8b,76,0a,8a,04
[925] 60130 DATA e8,00,00,5b,81,c3,0e,00,2e,88
[M45] 60140 DATA 07,eb,00,5b,58,55,cd,00,eb,18
[G45] 60150 DATA 90,fb,8b,c7,b4,00,83,c4,12,5d
[X25] 60160 DATA 1f,07,83,c4,06,5d,8b,76,08,89
[P45] 60170 DATA 04,eb,2a,90,5d,8b,76,1a,88,24
[L15] 60180 DATA 8b,76,18,88,04,8b,76,16,88,3c

```



```
[L15] 60190 DATA 8b,76,14,88,1c,8b,76,12,88,2c
[925] 60200 DATA 8b,76,10,88,0c,8b,76,0e,88,34
[A45] 60210 DATA 8b,76,0c,88,14,b8,00,00,8e,d8
[P25] 60220 DATA 58,a3,92,00,58,a3,90,00,8c,c0
[3P5] 60230 DATA 8e,d8,5f,8b,e7,5d,ca,16,00,ENDE
```

Listing 2. Das Hauptprogramm »interbas.bas« bildet die Grundlage für weitere Anwendungen

```
[YA0] 1 '
[0B9] 2 'Fenstertechnik unter GWBasic in der DOS intern
ational
[G94] 3 ' von Markus Zietlow
[AB0] 7 '
[H03] 8 'Initialisierung
[EB0] 9 '
[C53] 10 DEFINT A-Z: KEY OFF
[X72] 20 RESTORE 60030
[NWD] 30 READ CODE$: IF CODE$ <> "ENDE" THEN INTERRUPT$
= INTERRUPT$ + CHR$(VAL("&h" + CODE$)): GOTO 30
[AA1] 40 DEF SEG
[7M9] 50 FZEIGER = 1: LGRENZE = 1: OGRENZE = 1: UGRENZE
= 25: FALSE = 0
[98H] 60 FGRENZE = 4000: DIM FFELD$(10000): FOR Q% = 1
TO 10000: FFELD$(Q%) = " ": NEXT Q%: FOR Q% = 1 TO
0 4: FFELD$(Q%) = "0": NEXT Q%
[D73] 70 STAPEL$ = SPACE$(80).
[PS7] 80 AH = 0: AL = 0: BH = 0: BL = 0: CH = 0: CL = 0
: DH = 0: DL = 0
[V84] 90 RUECKGABEPARAMETER = 255
[H00] 100 '
[KZ4] 110 'Anfang des Demo-Programms
[6H0] 120 '
[H41] 130 CLS
[9X3] 140 FOR ZAEHLER% = 1 TO 25
[DS4] 150 PRINT STRING$(80, 177);
[YN2] 160 NEXT ZAEHLER%
[TUD] 170 LINKS% = 3: RECHTS% = 50: OBEN% = 2: UNTEN% =
8: TITEL$ = "Erstes Demo-Fenster": GOSUB 50030
[AKE] 180 LINKS% = 20: RECHTS% = 60: OBEN% = 5: UNTEN%
= 15: TITEL$ = "Zweites Demo-Fenster": GOSUB 50030
[HHE] 190 LINKS% = 54: RECHTS% = 77: OBEN% = 4: UNTEN%
= 21: TITEL$ = "Drittes Demo-Fenster": GOSUB 50030
[VAE] 200 LINKS% = 2: RECHTS% = 30: OBEN% = 7: UNTEN% =
18: TITEL$ = "Viertes Demo-Fenster": GOSUB 50030
[ZRE] 210 LINKS% = 4: RECHTS% = 75: OBEN% = 11: UNTEN%
= 25: TITEL$ = "Fünftes Demo-Fenster": GOSUB 50030
[AS3] 220 FOR ZAEHLER% = 1 TO 4
[PDP] 230 TEXT$ = "***** Fenster und Pull-Down-Menüs
sind mit der Basic-Erweiterung aus der DOS Inter
national kein Problem mehr *****": GOSUB 52030
[UN2] 240 NEXT ZAEHLER%
[1P4] 250 F.Y% = 10: F.X% = 8: GOSUB 51030
[ZFD] 260 TEXT$ = "Drücken Sie bitte eine Taste für die
Menü-Demonstration!": GOSUB 52030
[E04] 270 WHILE INKEY$ = " ": WEND
[TVB] 280 LINKS% = 32: RECHTS% = 48: OBEN% = 10: UNTEN%
= 20: TITEL$ = "Menü": GOSUB 50030
[HC2] 290 RESTORE 450
[CS3] 300 FOR ZAEHLER% = 1 TO 7
[0C2] 310 READ PUNKT$
[2W5] 320 F.Y% = ZAEHLER%: F.X% = 4: GOSUB 51030
[FT4] 330 TEXT$ = PUNKT$: GOSUB 52030
[WN2] 340 NEXT ZAEHLER%
[D52] 350 GOSUB 51530
[R52] 360 GOSUB 53030
[K52] 370 GOSUB 50530
[U92] 380 IF M.PUNKT = 0 THEN F.Y% = 3: F.X% = 18: GOSU
B 51030: TEXT$ = "Sie haben die <ESC>-Taste gedrü
ckt!": GOSUB 52030 ELSE F.Y% = 3: F.X% = 18: GOSU
B 51030: TEXT$ = "Sie haben den Menüpunkt" + STR$
(M.PUNKT) + " ausgewählt!": GOSUB 52030
[XP4] 390 F.Y% = 8: F.X% = 6: GOSUB 51030
[FHD] 400 TEXT$ = "Drücken Sie bitte eine Taste für das
Ende der Demonstration": GOSUB 52030
[004] 410 WHILE INKEY$ = " ": WEND
[GS3] 420 FOR ZAEHLER% = 1 TO 5
[F82] 430 GOSUB 53030
[YN2] 440 NEXT ZAEHLER%
[VPB] 450 DATA "Auswahl", "erfolgt", "durch", "<RETURN
>", "Abbruch", "durch", "<ESC>"
[E31] 460 END
[YT0] 10000 '
[2N7] 10010 'Unterprogramm für den Interruptaufruf
[0U0] 10020 '
[7SC] 10030 INTERRUPT! = PEEK(VARPTR(INTERRUPT$) + 2) *
256 + PEEK(VARPTR(INTERRUPT$) + 1)
[LFF] 10040 CALL INTERRUPT!(AH, AL, BH, BL, CH, CL, DH, DL, NUM
MER.DES.INTERRUPTS, RUECKGABEPARAMETER, STAPEL$)
```

```
[HV1] 10050 RETURN
[6U0] 50000 '
[P87] 50010 'Unterprogramm, das ein Fenster öffnet
[8U0] 50020 '
[NL4] 50030 FCODE = FALSE: GOSUB 54030
[YF7] 50040 IF LINKS% < LGRENZE THEN FCODE = 1: GOTO 50
460
[6V7] 50060 IF RECHTS% > RGRENZE THEN FCODE = 2: GOTO 5
0460
[ZF7] 50070 IF OBEN% < OGRENZE THEN FCODE = 3: GOTO 504
60
[7K7] 50080 IF UNTEN% > UGRENZE THEN FCODE = 4: GOTO 50
460
[AL7] 50090 IF LINKS% >= RECHTS% THEN FCODE = 5: GOTO 5
0460
[YC7] 50100 IF OBEN% >= UNTEN% THEN FCODE = 6: GOTO 504
60
[DL7] 50110 IF UNTEN% - OBEN% < 4 THEN FCODE = 7: GOTO
50460
[6W7] 50120 IF RECHTS% - LINKS% < 2 THEN FCODE = 8: GOT
O 50460
[UD4] 50130 DIF% = RECHTS% - LINKS% - 1
[G9A] 50140 LAENGE = FZEIGER + (RECHTS% - LINKS% + 1) *
(UNTEN% - OBEN% + 1) + 7 + 4
[GH7] 50150 IF LAENGE > FGRENZE THEN FCODE = 9: GOTO 50
460
[7U5] 50160 FFELD$(FZEIGER + 4) = CHR$(LINKS%)
[326] 50170 FFELD$(FZEIGER + 5) = CHR$(RECHTS%)
[KM5] 50180 FFELD$(FZEIGER + 6) = CHR$(OBEN%)
[7V5] 50190 FFELD$(FZEIGER + 7) = CHR$(UNTEN%)
[6YT] 50200 WERT = VAL("&h" + FFELD$(FZEIGER) + FFELD$(
FZEIGER + 1) + FFELD$(FZEIGER + 2) + FFELD$(FZEIG
ER + 3)): IF WERT <> 0 THEN FFELD$(WERT + 8) = CH
R$(F.X): FFELD$(WERT + 9) = CHR$(F.Y)
[AR2] 50210 F.X = 1: F.Y = 1
[AF4] 50220 FOR Y% = OBEN% TO UNTEN%
[ET4] 50230 FOR X% = LINKS% TO RECHTS%
[3LE] 50240 FFELD$(FZEIGER + 10 + (Y% - OBEN%) * (R
ECHTS% - LINKS% + 1) + (X% - LINKS%)) = CHR$(SCRE
EN(Y%, X%))
[K22] 50250 NEXT X%
[4S1] 50260 NEXT Y%
[AAA] 50270 NZEIGER = FZEIGER + (RECHTS% - LINKS% + 1)
* (UNTEN% - OBEN% + 1) + 10
[ZEL] 50280 ZAHL$ = LEFT$("0000", 4 - LEN(HEX$(FZEIGER
))) + HEX$(FZEIGER): FOR Q% = 1 TO 4: FFELD$(NZEIG
ER + Q% - 1) = MID$(ZAHL$, Q%, 1): NEXT Q%
[RN3] 50290 FZEIGER = NZEIGER
[9MC] 50300 IF LEN(TITEL$) > DIF% THEN NTITEL$ = LEFT$(
TITEL$, DIF%) ELSE NTITEL$ = TITEL$
[5C5] 50310 LFREI% = (DIF% - LEN(NTITEL$)) / 2
[946] 50320 RFREI% = DIF% - LEN(NTITEL$) - LFREI%
[PV3] 50330 LOCATE OBEN%, LINKS%
[468] 50340 PRINT CHR$(201); STRING$(DIF%, CHR$(205));
CHR$(187);
[P74] 50350 LOCATE OBEN% + 1, LINKS%
[P4C] 50360 PRINT CHR$(186); STRING$(LFREI%, 32); NTITE
L$: STRING$(RFREI%, 32); CHR$(186);
[V74] 50370 LOCATE OBEN% + 2, LINKS%
[E78] 50380 PRINT CHR$(199); STRING$(DIF%, CHR$(196));
CHR$(182);
[975] 50390 FOR Y% = OBEN% + 3 TO UNTEN% - 1
[VA3] 50400 LOCATE Y%, LINKS%
[228] 50410 PRINT CHR$(186); STRING$(DIF%, CHR$(32));
CHR$(186);
[8S1] 50420 NEXT Y%
[K64] 50430 LOCATE UNTEN%, LINKS%
[968] 50440 PRINT CHR$(200); STRING$(DIF%, CHR$(205));
CHR$(188);
[2KA] 50450 LINKS = LINKS%: RECHTS = RECHTS%: OBEN = OB
EN%: UNTEN = UNTEN%
[WU0] 50460 '
[985] 50470 IF FCODE THEN GOSUB 53530: END
[CW1] 50480 RETURN
[UU0] 50500 '
[E27] 50510 'Unterprogramm, das ein Fenster löscht
[WU0] 50520 '
[KW2] 50530 FCODE = FALSE
[7FN] 50540 WERT = VAL("&h" + FFELD$(FZEIGER) + FFELD$(
FZEIGER + 1) + FFELD$(FZEIGER + 2) + FFELD$(FZEIG
ER + 3)): IF WERT = 0 THEN FCODE = 10: GOSUB 5353
0: END
[6X4] 50550 FOR Y% = OBEN + 3 TO UNTEN - 1
[DM3] 50560 LOCATE Y%, LINKS + 1
[0H6] 50570 PRINT STRING$(RECHTS - LINKS - 1, 32);
[KS1] 50580 NEXT Y%
[LM1] 50590 F.X = 1
[GM1] 50600 F.Y = 1
[DW1] 50610 RETURN
[9U0] 51000 '
[E6C] 51010 'Unterprogramm, das den Cursor innerhalb ei
nes Fensters positioniert
[BU0] 51020 '
[1W2] 51030 FCODE = FALSE
[1PM] 51040 WERT = VAL("&h" + FFELD$(FZEIGER) + FFELD$(
FZEIGER + 1) + FFELD$(FZEIGER + 2) + FFELD$(FZEIG
ER + 3)): IF WERT = 0 THEN FCODE = 10: GOTO 51090
```



```

[X4A] 51050 IF (F.Y% > UNTEN - OBEN - 3) OR (F.Y% < 1)
      THEN FCODE = 11: GOTO 51090
[BZA] 51060 IF (F.X% > RECHTS - LINKS - 1) OR (F.X% < 1)
      THEN FCODE = 12: GOTO 51090
[UW5] 51070 LOCATE OBEN + F.Y% + 2, LINKS + F.X%
[TW3] 51080 F.X = F.X%: F.Y = F.Y%
[KU0] 51090 '
[T75] 51100 IF FCODE THEN GOSUB 53530: END
[WV1] 51110 RETURN
[XU0] 51500 '
[B67] 51510 'Unterprogramm, das ein Menü darstellt
[ZU0] 51520 '
[NW2] 51530 FCODE = FALSE
[Z27] 51540 IF (RECHTS - LINKS < 7) THEN FCODE = 14: GO
      TO 51680
[2K2] 51550 Y% = OBEN + 3
[3V0] 51560 '
[ERD] 51570 ALTS = CHR$(SCREEN(Y%, LINKS + 1)) + CHR$(S
      CREEN(Y%, LINKS + 2)) + CHR$(SCREEN(Y%, LINKS + 3
      ))
[M6F] 51580 ALTS = ALTS + CHR$(SCREEN(Y%, RECHTS - 3))
      + CHR$(SCREEN(Y%, RECHTS - 2)) + CHR$(SCREEN(Y%,
      RECHTS - 1))
[PS5] 51590 LOCATE Y%, LINKS + 1: PRINT " » ";
[F66] 51600 LOCATE Y%, RECHTS - 3: PRINT " « ";
[2V0] 51610 '
[413] 51620 TASTE$ = INKEY$
[6CW] 51630 IF TASTE$ = CHR$(0) + CHR$(72) THEN LOCATE
      Y%, LINKS + 1: PRINT LEFT$(ALTS, 3); : LOCATE Y%,
      RECHTS - 3: PRINT RIGHT$(ALTS, 3); : Y% = Y% - 1
      : WHILE Y% = OBEN + 2: Y% = UNTEN - 1: WEND: GOTO 515
      60
[EZV] 51640 IF TASTE$ = CHR$(0) + CHR$(80) THEN LOCATE
      Y%, LINKS + 1: PRINT LEFT$(ALTS, 3); : LOCATE Y%,
      RECHTS - 3: PRINT RIGHT$(ALTS, 3); : Y% = Y% + 1
      : WHILE Y% = UNTEN: Y% = OBEN + 3: WEND: GOTO 515
      60
[8YP] 51650 IF TASTE$ = CHR$(13) THEN LOCATE Y%, LINKS
      + 1: PRINT LEFT$(ALTS, 3); : LOCATE Y%, RECHTS -
      3: PRINT RIGHT$(ALTS, 3); : M.PUNKT = Y% - OBEN -
      2: GOTO 51680
[PDN] 51660 IF TASTE$ = CHR$(27) THEN LOCATE Y%, LINKS
      + 1: PRINT LEFT$(ALTS, 3); : LOCATE Y%, RECHTS -
      3: PRINT RIGHT$(ALTS, 3); : M.PUNKT = 0: GOTO 516
      80
[T32] 51670 GOTO 51610
[9V0] 51680 '
[P85] 51690 IF FCODE THEN GOSUB 53530: END
[LW1] 51700 RETURN
[CU0] 52000 '
[GD9] 52010 'Unterprogramm, das einen Text im Fenster d
      arstellt
[EU0] 52020 '
[4W2] 52030 FCODE = FALSE
[UH5] 52040 IF LEN(TEXT$) = 0 THEN GOTO 52150
[ZR4] 52050 FOR AGAIN = 1 TO LEN(TEXT$)
[Z59] 52060 IF ASC(MID$(TEXT$, AGAIN, 1)) < 32 THEN A
      GAIN = LEN(TEXT$)
[4B2] 52070 NEXT AGAIN
[Z08] 52080 IF AGAIN <= LEN(TEXT$) THEN FCODE = 13: GOT
      O 52150
[3S4] 52090 FOR AGAIN = 1 TO LEN(TEXT$)
[BM5] 52100 LOCATE OBEN + F.Y + 2, LINKS + F.X
[625] 52110 PRINT MID$(TEXT$, AGAIN, 1);
[AK2] 52120 F.X = F.X + 1
[D0M] 52130 IF (F.X > RECHTS - LINKS - 1) THEN F.X =
      1: F.Y = F.Y + 1: IF (F.Y > UNTEN - OBEN - 3) THE
      N FARBE% = 0: GOSUB 52530: F.Y = UNTEN - OBEN - 3
[5B2] 52140 NEXT AGAIN
[NU0] 52150 '
[285] 52160 IF FCODE THEN GOSUB 53530: END
[5W1] 52170 RETURN
[C0V] 52500 '
[C77] 52510 'Unterprogramm, das ein Fenster scrollt
[2V0] 52520 '
[SW2] 52530 FCODE = FALSE
[DFN] 52540 WERT = VAL("&H" + FFELDS(FZEIGER) + FFELDS(
      FZEIGER + 1) + FFELDS(FZEIGER + 2) + FFELDS(FZEIG
      ER + 3)): IF WERT = 0 THEN FCODE = 10: GOSUB 5353
      0: END
[K03] 52550 AH = &H6: AL = &H1
[263] 52560 BH = FARBE%: BL = 0
[XD4] 52570 CH = OBEN + 2: CL = LINKS
[1V4] 52580 DH = UNTEN - 2: DL = RECHTS - 2
[T87] 52590 NUMMER.DES.INTERRUPTS = &H10: GOSUB 10030
[KW1] 52600 RETURN
[FU0] 53000 '
[BV9] 53010 'Unterprogramm, das das geöffnete Fenster s
      chließt
[HU0] 53020 '
[7W2] 53030 FCODE = FALSE
[VNM] 53040 WERT = VAL("&H" + FFELDS(FZEIGER) + FFELDS(
      FZEIGER + 1) + FFELDS(FZEIGER + 2) + FFELDS(FZEIG
      ER + 3)): IF WERT = 0 THEN FCODE = 10: GOTO 53230
[113] 53050 FZEIGER = WERT
[8H5] 53060 LINKS = ASC(FFELDS(FZEIGER + 4))

```

```

[NL5] 53070 RECHTS = ASC(FFELDS(FZEIGER + 5))
[WAS] 53080 OBEN = ASC(FFELDS(FZEIGER + 6))
[XH5] 53090 UNTEN = ASC(FFELDS(FZEIGER + 7))
[044] 53100 FOR Y% = OBEN TO UNTEN
[F83] 53110 LOCATE Y%, LINKS
[WAH] 53120 FOR Q% = 1 TO RECHTS% - LINKS% + 1: PRINT
      FFELDS(FZEIGER + 10 + (Y% - OBEN) * (RECHTS - LI
      NKS + 1) + Q% - 1); : NEXT Q%
[6S1] 53130 NEXT Y%
[B3L] 53140 WERT = VAL("&H" + FFELDS(FZEIGER) + FFELDS(
      FZEIGER + 1) + FFELDS(FZEIGER + 2) + FFELDS(FZEIG
      ER + 3)): IF WERT = 0 THEN GOTO 53210
[335] 53150 LINKS = ASC(FFELDS(WERT + 4))
[Z35] 53160 RECHTS = ASC(FFELDS(WERT + 5))
[RR4] 53170 OBEN = ASC(FFELDS(WERT + 6))
[V35] 53180 UNTEN = ASC(FFELDS(WERT + 7))
[3M4] 53190 F.X = ASC(FFELDS(WERT + 8))
[4M4] 53200 F.Y = ASC(FFELDS(WERT + 9))
[SU0] 53210 '
[NLL] 53220 WERT = VAL("&H" + FFELDS(FZEIGER) + FFELDS(
      FZEIGER + 1) + FFELDS(FZEIGER + 2) + FFELDS(FZEIG
      ER + 3)): IF WERT = 0 THEN F.X = 0: F.Y = 0
[UU0] 53230 '
[785] 53240 IF FCODE THEN GOSUB 53530: END
[AW1] 53250 RETURN
[3V0] 53500 '
[573] 53510 'Fehler-Routine
[5V0] 53520 '
[AB6] 53530 PRINT : PRINT "Fehler"; FCODE; "; ";
[VDA] 53540 IF FCODE = 1 THEN PRINT "Linke Grenze des F
      ensters zu klein"
[8EA] 53550 IF FCODE = 2 THEN PRINT "Rechte Grenze des
      Fensters zu groß"
[TDA] 53560 IF FCODE = 3 THEN PRINT "Obere Grenze des F
      ensters zu klein"
[AGA] 53570 IF FCODE = 4 THEN PRINT "Untere Grenze des
      Fensters zu groß"
[Y79] 53580 IF FCODE = 5 THEN PRINT "Linke Grenze >= Re
      chte Grenze"
[K99] 53590 IF FCODE = 6 THEN PRINT "Obere Grenze >= Un
      tere Grenze"
[RS9] 53600 IF FCODE = 7 THEN PRINT "Obere Grenze - Unt
      ere Grenze < 4"
[PG9] 53610 IF FCODE = 8 THEN PRINT "Rechte Grenze - Li
      nke Grenze < 2"
[XNB] 53620 IF FCODE = 9 THEN PRINT "Stapel für Fenster
      mit"; FGRENZE; "zu klein"
[TM9] 53630 IF FCODE = 10 THEN PRINT "Alle Fenster sind
      geschlossen"
[9VA] 53640 IF FCODE = 11 THEN PRINT "Y-Koordinate auß
      erhalb des Fensters"
[CVA] 53650 IF FCODE = 12 THEN PRINT "X-Koordinate auß
      erhalb des Fensters"
[RXA] 53660 IF FCODE = 13 THEN PRINT "Text enthält ASCI
      I-Zeichen kleiner 32"
[R7B] 53670 IF FCODE = 14 THEN PRINT "Fenster für Menü
      zu schmal (Breite<6)"
[XW1] 53680 RETURN
[KU0] 54000 '
[PV9] 54010 'Unterprogramm, das die System-Konfiguratio
      n ausliest
[MU0] 54020 '
[T47] 54030 AH=&HF:AL=&H0:NUMMER.DES.INTERRUPTS=&H10
[M92] 54040 GOSUB 10000
[608] 54050 RGRENZE = (AH * 256 + AL) / 256 'Anzahl der
      Spalten
[SU0] 54060 '
[7W1] 54070 RETURN
[8U0] 60000 '
[NH7] 60010 'Daten für das Maschinensprachprogramm
[AU0] 60020 '
[C65] 60030 DATA 55,8b,ec,8b,fc,8b,76,06,8a,04
[M15] 60040 DATA b4,00,46,03,04,48,8b,e0,57,b8
[705] 60050 DATA 00,00,0e,d8,a1,90,00,50,a1,92
[RY4] 60060 DATA 00,50,e8,00,00,5b,81,c3,56,00
[105] 60070 DATA 89,1e,90,00,0e,58,a3,92,00,8c
[G55] 60080 DATA c0,8e,d8,8b,76,1a,8a,24,8b,76
[345] 60090 DATA 18,8a,04,8b,76,16,8a,3c,8b,76
[F55] 60100 DATA 14,8a,1c,8b,76,12,8a,2c,8b,76
[945] 60110 DATA 10,8a,0c,8b,76,0e,8a,34,8b,76
[115] 60120 DATA 0c,8a,14,50,53,8b,76,0a,8a,04
[925] 60130 DATA e8,00,00,5b,81,c3,0e,00,2e,88
[M45] 60140 DATA 07,eb,00,5b,58,55,cd,00,eb,18
[G45] 60150 DATA 90,fb,8b,c7,b4,00,83,c4,12,5d
[X25] 60160 DATA 1f,07,83,c4,06,5d,8a,76,08,89
[P45] 60170 DATA 04,eb,2a,90,5d,8b,76,1a,88,24
[L15] 60180 DATA 8b,76,18,88,04,8b,76,16,88,3c
[L15] 60190 DATA 8b,76,14,88,1c,8b,76,12,88,2c
[925] 60200 DATA 8b,76,10,88,0c,8b,76,0e,88,34
[A45] 60210 DATA 8b,76,0c,8b,14,b8,00,00,8e,d8
[P25] 60220 DATA 58,a3,92,00,58,a3,90,00,8c,c0
[3P5] 60230 DATA 8e,d8,5f,8b,e7,5d,c4,16,00,ENDE

```

Listing 3. Die Befehlerweiterung "erweiter.bas" zeigt Ihnen anhand der Window-Routinen, wie Sie mit geringem Einsatz aufwendige Effekte erzielen

PC-SPIELEBOX NO.1

BIMBO II

Das Geschicklichkeitsspiel für den PC überhaupt!

Wie in einem Squashcourt spielen Sie einen Ball von rechts nach links durch einen auf drei Seiten geschlossenen Bildschirm. Auf der offenen Seite müssen Sie mit Ihrem Schläger verhindern, daß der Ball ins Aus geht. Im Spielfeld selbst befinden sich unterschiedliche Hindernisse, die jedes für sich, unterschiedliche Auswirkungen auf den weiteren Spielverlauf haben. So gibt es Blöcke, die dem Spieler, sobald sie getroffen wurden, einen Extraball gutschreiben. Andere wiederum ziehen einen Ball ab, zählen Bonus, verschnellern das Spiel und und und...

Ist es Ihnen gelungen, alle Hindernisse zu beseitigen, gelangen Sie in das nächste Bild von Bimbo, das Sie wiederum mit neuen Hinderniskonstellationen konfrontiert.

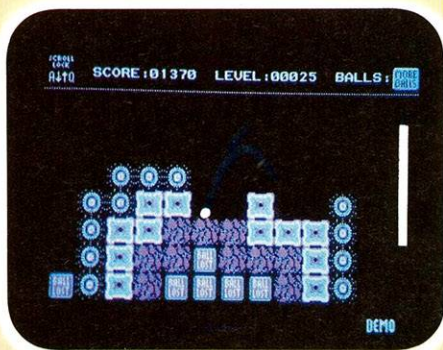
Aber Bimbo ist mehr als nur ein Spiel. Mittels des zum Programm gehörenden Editors sind Sie in der Lage, sich genau die Bilder zusammenzusetzen, die Sie gerne spielen möchten.

BOUNCER

Bouncer versetzt Sie hinter das Lenkrad eines ganz besonderen Automobils. Das Bounce'O'mobil kann springen. Daß es dies nicht nur aus Spaß an der Freude tut, wird spätestens beim ersten Felsbrocken klar, der Ihnen den Weg versperrt. Nun sind Sie gefordert, überspringen Sie das Hindernis. Jedoch werden Sie nicht nur durch herumliegende Findlinge gehandicapt. So manche Brücke, die Sie auf Ihrer Fahrt überqueren müssen, ist eingestürzt, tollwütige Vögel versuchen, sich aus der Luft auf Sie zu stürzen, Straßensperren blockieren den Weg, und der rachsüchtige Pilot einer Sportmaschine stellt Ihnen nach.

Wird es Ihnen gelingen, die Straße bis zu ihrem Ende zu fahren?

Bouncer, die PC-Variante eines erfolgreichen Arcade Automaten, wurde 100% in Maschinencode geschrieben.



Best.-Nr. 5 1/4" Disk. 129 Best.-Nr. 3 1/2" Disk. 185
Für MS-DOS erhältlich

49,-DM*

PC-SPIELEBOX NO.2

JACKY

Hier erwartet Sie ein Geschicklichkeitsspiel der besonderen Art.

Sie steuern einen kleinen Roboter, der von Ebene zu Ebene springen kann. Allerdings stehen diese unter Strom, so daß Jacky versuchen muß, durch die Lücken zu springen, ohne die Ebenen von unten zu berühren. Dabei gilt es natürlich, möglichst nicht die vielen Monster zu berühren, die Jacky das Leben schwer machen. Auf der anderen Seite können Sie auf Ihrem Weg nach oben eine ganze Reihe von Boni einsammeln, die Ihren Score gehörig aufpolieren können. Programmiert von Hartmut Pfarr, dem bekannten DMV-Autor, verspricht Jacky ein echter Renner zu werden. Versäumen Sie also auf keinen Fall die aufregenden Abenteuer des kleinen Roboters Jacky.

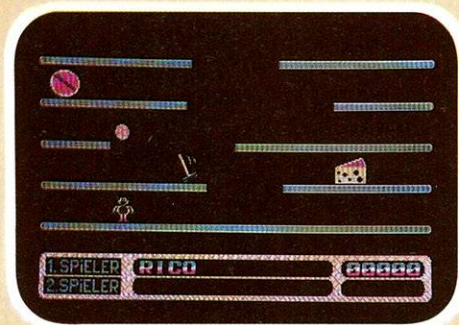
Die Abenteuer des kleinen Jacky sind im System MS-DOS erhältlich.

STARVISION

Ein exzellentes Shoot'em up auf Ihrem PC!

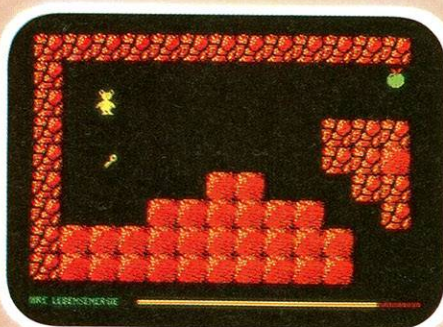
Besteigen Sie Ihr Schiff und retten Sie den Mond. Außerirdische Kreaturen haben sich dort eingenistet und behindern die Aktivitäten der Menschen. Deshalb ist es jetzt Ihre Aufgabe, den Mond von diesen Wesen zu befreien. Die größte Hilfe bei Ihrer Aufgabe ist Ihr kleines Raumschiff, daß Sie im Verlauf dieses Spieles mit zahlreichen Extrateatues ausrüsten können. Trotzdem keine einfache Aufgabe, da die Außerirdischen bereits mehrere Stützpunkte eingerichtet haben. Wenn Sie den ersten Sektor gesäubert haben, bekommen Sie einen Bonus und können sich am nächsten versuchen. Starvision verspricht somit viele Stunden spannender Unterhaltung.

Starvision ist im System MS-DOS erhältlich.



Best.-Nr. 5 1/4" Disk. 1291 Best.-Nr. 3 1/2" Disk. 1292

49,-DM*



5 1/4" Disk. Best.-Nr. 186
3 1/2" Disk. Best.-Nr. 187

49,-DM*

Underground Strategie & Action für Ihren PC!

Wer denkt, daß unter der Erde tiefer Friede herrscht, der irrt sich gewaltig. Unbemerkt von uns, den oberirdischen Betrachtern, spielt sich unterhalb der Erdoberfläche so manches Drama im Verborgenen ab.

Underground, so der Titel des neuesten Hits von DMV, beleuchtet die Vorgänge unter der Erde. Begleiten Sie den kleinen Maulwurf, den Star dieses Programms, bei seinem heroischen Kampf gegen seine greulichen Feinde, und helfen Sie ihm, den Ausweg aus den finsternen und lichtlosen Katakomben des Schreckens zu finden.

Doch erwarten Sie keine langweiligen Grabespiele, wie man bei einem Maulwurf vermuten könnte! Spätestens, seit prominente Frösche Eigentumswohnungen bewohnen, haben sich die Standards im Tierleben durchgreifend geändert.

Maulwürfe, die up to date sind, haben Ihr Labyrinth beispielsweise mit hochmodernen Beam-Stationen ausgerüstet, die einen zeitsparenden Transport zwischen abgelegenen Teilen des Systems ermöglichen.

In Actioneinlagen müssen Feinde in ihre Schranken verwiesen werden, die den kleinen Maulwurf zum Fressen gern haben. Eine Landkarte erleichtert die Orientierung innerhalb der jeweiligen Labyrinth-Ebene, diese Karte muß allerdings erst gefunden werden. Bei der Suche ist so mancher Gegenstand zu finden, der das Abenteuer erleichtern kann, so gibt es Nahrung, die die schwindenden Kräfte wieder aufstockt, eine Fackel, mit der lichtscheues Gesindel vertrieben werden kann, kleine Items, die den Maulwurf für kurze Zeit zum unbesiegbaren Superhelden machen, und, und, und.

Wenn Sie Streit mit potentiellen Gegnern anfangen, können Sie Ihre Zielgenauigkeit im Steinwurf unter Beweis stellen, um Ihre Feinde Mores zu lehren.

Wird es Ihnen gelingen, den kleinen Maulwurf aus den Katakomben des Schreckens zu befreien?

Die Abenteuer des kleinen Maulwurfes sind für das System MS-DOS (CGA) erhältlich.

Jetzt neu

KNOW-PC

Das Spiel für die ganze Familie!

Mehr als nur ein Spiel KNOW-PC

- kann eine unbegrenzte Anzahl von Fragen und Antworten verwalten • bietet die Möglichkeit, eigene Fragen in einem selbstgewählten Wissensgebiet einzugeben, und zwar mit einem komfortablen Editor • ist vollkommentenue gesteuert • besitzt eine Supergrafik • stellt zu jeder Frage wahlweise 5 oder 1 Antwort zur Auswahl • verhindert, daß eine richtig beantwortete Frage nochmals erscheint • gibt alle Fragen und Antworten sowie sämtliche Aufforderungen für jeden Spieler und für jedes Wissensgebiet mit Angabe der jeweiligen Platzierung aus • ist spielbar mit 1 – 4 Einzelspielern oder in Gruppen • kann man mit der ganzen Familie spielen, da jeder eine Mindestchance von 1 : 5 hat • muß man mit Strategie spielen, da der Beste in jedem Wissensgebiet am Ende noch Zusatzpunkte bekommt • verlangt keinerlei Programmierkenntnisse • wird mit 13 verschiedenen Wissensgebieten ausgeliefert • kann auch für andere Zwecke Verwendung finden, z.B. als Vokabeltrainer.

Was ist KNOW-PC?

KNOW-PC ist ein Frage- und Antwortspiel der Extraklasse, denn es bietet Ihnen nicht nur kurzweilige Unterhaltung, sondern ist darüber hinaus hervorragend dazu geeignet, eigene Fragen einzugeben, mit denen Sie Ihre Familie, Ihren Skatverein, Ihre Mitschüler, Kegelbrüder, Arbeitskollegen, Freunde usw. überraschen können.

Konfiguration:

PC XT/AT mit 512 kByte oder mehr, 1 Diskettenlaufwerk oder Festplatte, MS-DOS ab Version 2.0 oder größer, für Farbgrafik, Monochrom, Hercules oder EGA.

* Unabhängig von der Anzahl der bestellten Programme berechnen wir für das Inland 3,- DM bzw. für das Ausland 5,- DM Porto und Verpackung. – Unverbindliche Preisempfehlung – Bitte benutzen Sie die Bestellkarte



Best.-Nr. 5 1/4" Disk. 161
Best.-Nr. 3 1/2" Disk. 162

49,-DM*



DMV GmbH • Postfach 250 • 3440 Eschwege

GW-Basic intern

Wenn Sie mit dem GW-Basic-Interpreter arbeiten, nehmen Sie nicht wahr, wie komplex die dabei ablaufenden Vorgänge sind. Dieser Beitrag läßt Sie ein wenig hinter die Kulissen der Arbeitsweise des Basic-Interpreters schauen.

Der GW-Basic-Interpreter stellt für das Basic-Programm inklusive der Variablen und des Stapels nur 64 KByte zur Verfügung. Am Anfang dieses Speicherbereichs steht der Programmtext, allerdings nicht in Form von ASCII-Zeichen. Vielmehr wandelt der Interpreter die eingegebenen Zeilen in die sogenannte Token-Schreibweise um, das heißt, für jedes erkannte Basic-Schlüsselwort (»token«), wie zum Beispiel »print« oder »goto«, setzt er einen speziellen Code ein.

Auf diese Weise schlägt der Interpreter zwei Fliegen mit einer Klappe. Er spart kostbaren Speicherplatz, da für jeden Basic-Befehl nur ein oder zwei Byte nötig sind, gegenüber bis zu zehn Zeichen im ASCII-Format. Des weiteren beschleunigt dies die Ausführung des Basic-Programms, denn der Interpreter muß nicht bei jedem Aufruf des Programms einen ASCII-Text nach Schlüsselworten untersuchen, sondern kann gleich die zu den entsprechenden Token gehörenden Routinen ausführen.

Dieses Token-Format im Speicher entspricht demjenigen, in dem der GW-Basic-Interpreter seine Program-

me auch auf Diskette beziehungsweise Festplatte schreibt – wenn Sie nicht durch den Parameter »a« ausdrücklich das ASCII-Format gewünscht haben.

Die Struktur einer Basic-Token-Datei ist an die Speicherverwaltung angepaßt

Das erste Byte eines Basic-Programms in Token-Form ist immer FFhex. Daran erkennt der Interpreter beim Laden einer Datei, daß diese bereits in Token-Form vorliegt und nicht mehr in Schlüsselworte zerlegt werden muß. Das zweite Byte ist für besondere Zwecke reserviert.

Nach diesen zwei Byte beginnt das eigentliche Basic-Programm. Zunächst folgt die Zeilennummer der ersten Programmzeile, die durch das Kennbyte 0Fhex oder 12hex eingeleitet wird. Dieses Kennbyte weist den Basic-Interpreter an, die nächsten zwei Byte als 16-Bit-Integerzahl zu verwenden. Der Anfang der Zeile mit der Nummer 10 steht im Speicher beziehungsweise auf Diskette als

12 A0 00

Aus der Verwendung einer 16-Bit-Zahl für die Zeilennummer ergibt sich auch die Beschränkung auf maximal 65535 Zeilennummern. Der für Basic-Programm und Variablen zur Verfügung stehende Speicherbereich von nur 64 KByte setzt der Länge Ihrer Programme allerdings viel früher Grenzen.

Direkt auf die ersten drei Byte für die Zeilennummer folgt die erste Basic-Anweisung dieser Zeile als Token. Dabei gibt es drei Arten von Token:

– Die 1-Byte-Token bei denen das Token durch ein einziges Byte dargestellt wird. Darunter fallen fast alle unbedingt notwendigen Basic-Befehle, wie zum Beispiel »print« (91hex). Der Wert eines 1-Byte-Token liegt immer zwischen 80hex und FFhex. Tabelle 1 zeigt alle Token, die durch ein Byte repräsentiert werden.

– Die 2-Byte-Token, die mit dem Kennbyte FEhex beginnen. Ein solches Token speichert der Interpreter in zwei aufeinanderfolgenden Speicherstellen ab, wobei in der ersten immer der Wert FEhex steht und in der zweiten ein Wert aus Tabelle 2. Unter diese Befehle fallen zum Beispiel die Grafikbefehle und die Netzwerk-Operationen.

– Die 2-Byte-Token, die mit dem Kennbyte FFhex beginnen. Diese Schlüsselwörter werden genauso behandelt wie die 2-Byte-Token mit dem Kennbyte FEhex, nur hat hierbei das erste Byte den Wert FFhex. Fast alle Funktionen fallen unter diese Gruppe. Tabelle 3 zeigt alle Schlüsselwörter, die dazugehören.

Damit der Interpreter unterscheiden kann, ob es sich um ein 1-Byte oder um ein 2-Byte langes Token handelt, folgt auf das eigentliche Token immer ein Leerzeichen (ASCII 32 = 20hex). Bei den 2-Byte-Token mit Kennbyte FEhex ist dies eigentlich

Token	Befehl	Token	Befehl	Token	Befehl	Token	Befehl
80	?	A0	WIDTH	C0	CLS	E2	?
81	END	A1	ELSE	C1	MOTOR	E3	?
82	FOR	A2	TRON	C2	BSAVE	E4	?
83	NEXT	A3	TROFF	C3	BLOAD	E5	?
84	DATA	A4	SWAP	C4	SOUND	E6	>
85	INPUT	A5	ERASE	C5	BEEP	E7	=
86	DIM	A6	EDIT	C6	PSET	E8	<
87	READ	A7	ERROR	C7	PRESET	E9	+
88	LET	A8	RESUME	C8	SCREEN	EA	.
89	GOTO	A9	DELETE	C9	KEY	EB	*
8A	RUN	AA	AUTO	CA	LOCATE	EC	/
8B	IF	AB	RENUM	CB	?	ED	↑
8C	RESTORE	AC	DEFSTR	CC	TO	EE	AND
8D	GOSUB	AD	DEFINT	CD	THEN	EF	OR
8E	RETURN	AE	DEFSNG	CE	TAB(	F0	XOR
8F	REM	AF	DEFDBL	CF	STEP	F1	EQV
90	STOP	B0	LINE	D0	USR	F2	IMD
91	PRINT	B1	WHILE	D1	FN	F3	MOD
92	CLEAR	B2	WEND	D2	SPC(	F4	?
93	LIST	B3	CALL	D3	NOT	F5	?
94	NEW	B4	?	D4	ERL	F6	?
95	ON	B5	?	D5	ERR	F7	?
96	WAIT	B6	?	D6	STRING\$	F8	?
97	DEF	B7	WRITE	D7	USING	F9	?
98	POKE	B8	OPTION	DA	VARPTR	FA	?
99	CONT	B9	RANDOMIZE	DB	CSRLEN	FB	?
9A	?	BA	OPEN	DC	POINT	FC	?
9B	?	BB	CLOSE	DD	OFF	FD	?
9C	OUT	BC	LOAD	DE	INKEY\$	FE	- - -
9D	LPRINT	BD	MERGE	DF	?	FF	PEN
9E	LLIST	BE	SAVE	E0	?		
9F	?	BF	COLOR	E1	?		

Tabelle 1. Diese Schlüsselwörter werden durch 1-Byte-Token dargestellt

Token	Befehl	Token	Befehl
01	CVI	90	COM
02	CVS	91	CIRCLE
03	CVD	92	DRAW
04	MKI\$	93	PLAY
05	MKS\$	94	TIMER
06	MKD\$	95	ERDEV
0B	EXTERR	96	IOCTL
81	FILES	97	CHDIR
82	FIELD	98	MKDIR
83	SYSTEM	99	RMDIR
84	NAME	9A	SHELL
85	LSET	9B	ENVIRON
86	RSET	9C	VIEW
87	KILL	9D	WINDOW
88	PUT	9E	PMAP
89	GET	9F	PALETTE
8A	RESET	A0	LCOPY
8B	COMMON	A1	CALLS
8C	CHAIN	A5	PCOPY
8D	DAT\$	A7	LOCK
8E	TIM\$	A8	UNLOCK
8F	PAINT		

Tabelle 2. Ein 2-Byte-Token besteht aus dem Kennbyte FEhex und einem der folgenden Werte

nicht erforderlich, denn dem Wert FHex ist kein 1-Byte-Token zugeordnet, der Interpreter »weiß« also, daß er noch ein weiteres Byte hinzuziehen muß. Anders verhält es sich bei den 2-Byte-Token mit Kennbyte FFhex, denn FFhex entspricht ja gleichzeitig dem 1-Byte-Token »PEN« (siehe Tabelle 1). Die Basic-Zeile »10 PRINT« wird im Speicher also als

12 A0 00 91 20
abgelegt.

Zeichenketten legt der Interpreter im ASCII-Format im Speicher ab. Sie werden durch ein Anführungszeichen eingeleitet und abgeschlossen. Zwischen diesen beiden Anführungszeichen steht der Inhalt der Zeichenkette als reiner ASCII-Text. Rem-Zeilen beginnen mit dem Code für das Token »REM« (8Fhex), anschließend folgt der Kommentar als Zeichenkette, die mit dem Zeichen ASCII 0hex abschließt.

Daß Ihnen unter GW-Basic nur maximal 64 KByte Speicherplatz für das Programm und alle seine Variablen zur Verfügung stehen, liegt daran, daß alle diese Daten in einem einzigen Speichersegment untergebracht sind. Dieses Segment nennt man auch Basic-Daten-Segment. Die Segmentregister DS, SS und ES zeigen alle auf dieses Segment, das direkt auf das Codesegment mit dem Interpreter folgt.

Den Anfang des Basic-Daten-Segments bildet der Arbeitsbereich für den Interpreter. Dort befinden sich unter anderem die Pufferspeicher für die seriellen Schnittstellen und für den Zugriff auf Dateien. Auf diese Daten folgt Ihr Basic-Programm in der oben beschriebenen Form.

Der Speicherbereich nach dem Basic-Programm ist für die numerischen Variablen und Arrays vorgesehen. Am Ende des Basic-Daten-Segments steht der Basic-Stapel, wo unter anderem die Rücksprungadressen bei Gosub-Befehlen abgelegt werden. Zwischen Basic-Stapel und dem Speicherbereich für die Arrays liegt der Stringspeicher.

Der noch zur Verfügung stehende freie Speicher, auf den Sie mit der Funktion »fre(0)« hingewiesen werden, liegt zwischen dem Ende des Stringspeichers und dem Ende des Speicherbereichs für die Arrays. Die Speicherbereiche für die verschiedenen Daten, also Basic-Programm,

Wert	Variablentyp
2	Integer
3	String
4	einfachgenaue Realzahl
8	doppeltgenaue Realzahl

Tabelle 4. Die Schlüsselwerte zur Kennzeichnung des Variablentyps

Token	Funktion	Token	Befehl	Token	Befehl	Token	Befehl
81	LEFT\$	8B	EXP	95	ASC	9F	FIX
82	RIGHT\$	8C	COS	96	CHR\$	A0	PEN (Funktion)
83	MID\$	8D	TAN	97	PEEK	A1	STICK
84	SGN	8E	ATN	98	SPACE\$	A2	STRIG
85	INT	8F	FRE	99	OCT\$	A3	EOF
86	ABS	90	INP	9A	HEX\$	A4	LOC
87	SQR	91	POS	9B	LPOS	A5	LOF
88	RND	92	LEN	9C	CINT		
89	SIN	93	STR\$	9D	CSNG		
8A	LOG	94	VAL	9E	CDBL		

Tabelle 3. Die 2-Byte-Token, die mit dem Kennbyte FFhex beginnen, sind für die Schlüsselwörter der Funktionen reserviert

Stringspeicher und so fort, sind in Ihrer Größe nicht festgelegt. Sie passen sich dynamisch den jeweiligen Bedingungen an. Eine Ausnahme stellt der Basic-Stapel dar, denn seine Größe ist normalerweise auf 512 Byte festgelegt und läßt sich nur durch den Befehl »clear <n>« verändern. Ein Programm darf also entweder sehr lang sein oder sehr viele Variablen verwenden. Dabei belegen das eigentliche Programm, die numerischen Variablen und die Arrays Speicherplatz vom Anfang des Basic-Daten-Segments hin zu dessen Ende, also in aufsteigender Richtung. Der Stringspeicher startet am Ende des Basic-Daten-Segments und dehnt sich absteigend aus.

Wenn das Ende des Stringspeichers auf das Ende des Array-Speichers trifft, ist kein freier Speicher mehr im Basic-Daten-Segment verfügbar und der Interpreter meldet den Fehler »Out of Memory«.

Programm, Arrays, Strings und die übrigen Variablen liegen geordnet im Speicher

Die Speicherplätze für die Variablenspeicherung belegt der GW-Basic-Interpreter dynamisch, also während der Ausführung des Basic-Programms. Der Variablenbereich bleibt so lange unangetastet, wie keine neuen Programmzeilen eingegeben oder bestehende geändert werden. Sobald sich die Länge des Programmspeichers auch nur um ein Byte ändert, verlieren die Variablen Ihre Werte. Das ist auch der Grund, warum Sie sich nach dem Ablauf oder Abbruch eines Programms so lange die Variableninhalte durch Befehle im Direktmodus auflisten können, wie Sie keine Änderungen am Programm vornehmen.

Für jede Variable reserviert der Interpreter eine gewisse Anzahl von Byte, wobei diese sowohl von der Art der Variable als auch von der Länge des Variablennamens abhängt. Das erste Byte enthält dabei einen 8-Bit-Wert, der Auskunft über die Art der Variable gibt. Dabei sind die Werte aus Tabelle 4 zulässig.

Die nächsten zwei Byte nehmen die ersten beiden Zeichen des Variablennamens auf, und Byte drei enthält die Anzahl der noch verbleibenden Zeichen des Variablennamens. Diese restlichen Zeichen legt der Interpreter dann beginnend bei Byte vier ab. Diese Aufspaltung des Variablennamens bringt erhebliche Geschwindigkeitsvorteile, denn GW-Basic muß sich bei der Suche nach den Speicherplätzen einer Variable zunächst nur um die ersten zwei Zeichen des Variablennamens kümmern. Nur wenn es mehrere Variablen gibt, bei denen der Variablenname in den ersten beiden Zeichen übereinstimmt, ist eine Untersuchung des gesamten Namens erforderlich.

Nach den restlichen Zeichen des Variablennamens folgt der Datenbereich dieser Variable. Er umfaßt zwei, drei, vier oder acht Byte, je nachdem ob es sich um eine Integerzahl, einen String oder um eine einfach- beziehungsweise um eine doppeltgenaue Realzahl handelt. Die Basic-Funktion »varptr« zeigt auf den Anfang dieses Wertebereichs. Wie Sie sehen, entspricht das Byte, das den Variablentyp angibt, genau der Anzahl Byte des Datenbereichs. Bei einer Integerzahl enthalten die beiden Byte das Low- und High-Byte des Zweierkomplements des Wertes, daher reicht der Wertebereich von -32768 bis + 32767.

Wie bereits erwähnt, legt der Interpreter Stringdaten in einem speziellen Speicherbereich unterhalb des Basic-Stapels ab. Im eigentlichen Variablenbereich steht daher nur ein Zeiger, der die Position und die Länge des Strings innerhalb des Stringspeichers angibt. Den drei Byte umfassenden Datenbereich eines Strings, der diese Informationen aufnimmt, nennt man »String Descriptor«.

Das erste Byte des String-Descriptors enthält die Länge des Strings in Byte. Im zweiten und dritten Byte steht die Adresse des Stringanfangs als Offset relativ zum Basic-Daten-Segment. Dabei nimmt das zweite Byte das niederwertige Byte der Adresse und das dritte das höherwertige Byte auf.

(Martin Hepp/db)

Tips und Tricks für Basic

Mit einfachen Aufrufen und kleinen aber ausgetüftelten Programmen können Sie sich selbst unter Basic Türen zur professionellen Programmgestaltung öffnen. Einige solcher Tricks werden hier vorgestellt und erläutert.

What time is it? – Zeitanzeige über Timer-Interrupt

Die Funktion »on timer(n) goto« in Basic scheint auf den ersten Blick sehr brauchbar zu sein. Nach ein wenig Praxis erkennt man jedoch schnell, daß die Verwendbarkeit sich in Grenzen hält.

Bildschirmdarstellungen in bestimmten Zeitabständen zu realisieren, bedeutet oft einen unverhältnismäßig hohen Programmieraufwand:

Die Position des Cursors und die augenblickliche Bildschirmfarbe müssen ermittelt und zwischengespeichert werden, wenn man sie in der Interruptroutine verändert. Befindet sich der Cursor in der rechten unteren Ecke, hilft selbst diese Maßnahme nichts, da im Falle seiner, wenn auch temporären, Verschiebung der Rechner den Schirm nicht mehr »sauber« scrollt. Als einzige Alternative bleibt, an diesen Werten nichts zu ändern. Es liegt also nahe, den Bildschirm in der Timer-Routine direkt zu beschreiben. Überdies ist es sinnvoll, sie nicht sekundenweise auszuführen, um vertretbare Ablaufzeiten zu gewährleisten.

Listing 1 zeigt die Zeit in Stunden und Minuten in der oberen linken Bildschirmecke an. Die Herculeskarte und reine Textgrafikkarten kennen den Beginn des Bildschirms im Speicher bei Segment B000hex. Bei der CGA-Karte steht er an der Segmentadresse B800hex. In der Zeile 10010 wird das Bildschirmsegment für den direkten Zugriff aktiviert.

Steckt eine Herculeskarte oder eine reine Textgrafikkarte im Slot, muß hier also »B000« statt »B800« angegeben sein. Wenn man im Hauptprogramm den Bildschirm löscht oder soviel am Bildschirm dargestellt hat, daß die Zeitinformation durch Scrollen nicht mehr in der linken

oberen Bildschirmecke steht, muß man die Routine mit der Anweisung

```
gosub 10000
```

direkt aufrufen. Ändert sich der Bildschirm ständig, bekommt das sekundenweise Bearbeiten der Zeitanzeige Sinn. Hierzu ändern Sie den Wert »60« in der Zeile 100 in eine »1«.

Es lohnt sich dann auch, die volle Zeit, also auch die Sekunden, anzuzeigen. Ersetzen Sie dazu dann den Endwert »8« in der Zählschleife in Zeile 10030 durch eine »14«. Damit erreichen Sie, daß die Variable »time\$« unbeschnitten in den Schirm geschrieben wird.

Ab Zeile 120 beginnt Ihr eigentliches Hauptprogramm. In Ihrem Programm müssen Sie jetzt nur noch darauf achten, daß Sie vor einer Segmentneudefinition mit dem Kommando »timer stop« den Timerinterrupt deaktivieren und diesen nach der zugehörigen Speicherzuweisung mit »timer on« wieder aktivieren. Das ist notwendig, weil sich genau nach einem Def-seg-Befehl ein Timerinterrupt ereignen könnte, wonach das Datensegment dann auf den Bildschirm zeigen würde. Nachfolgendes Beschreiben oder Auslesen aus dem nun falschen Segment zeigen dann »unerwünschte« Ergebnisse.

(db)

Spiel's noch einmal Sam! – Musik im Hintergrund

Wer seine Programme gerne mit musikalischen Effekten untermalen will, muß sich zumeist auf sehr kurze Einlagen beschränken. Der Basic-Befehlssatz erlaubt aber das Spielen beliebiger Tonsequenzen im Hintergrund.

Die Zeichenkette, die die Musikinformationen beinhaltet, muß, in »mundgerechte« Stücke zu etwa 16 Zeichen, in mehreren Zeichenketten aufgeteilt werden. Schnellere Abspielsequenzen dürfen dabei in größeren Strings stehen. Sie sind nämlich schneller bearbeitet.

Listing 2 zeigt ein Basicprogramm, das ein Musikstück in sechs verschiedenen Variationen im Hintergrund spielt, während das Hauptprogramm bis 9999 zählt. Die erste Anweisungszeile veranlaßt den Rechner, den Basic-eigenen Playinterrupt zuzulassen. Die Anweisung »on play(n) goto <Zeile>« legt fest, daß, sobald die automatische Hintergrundverarbeitung des letzten Play-Befehls nur noch n Befehlszeichen auszuführen hat, zu einer bestimmten Zeile verzweigt wird. Dazu ein Beispiel:

```
10 play on
20 play "cdefgabc"
30 on play(2) goto 60
40 if abbruch<>1 then goto 40
50 end
60 abbruch=1: return
```

Dieses Programm spielt die Tonleiter bis zur Note »b«, während die Verarbeitung der Zeile 40 andauert und »be-sucht« dann die Zeile 60, in der eine Variable als Abbruchmarke den Wert 1 bekommt. Aufgrund dessen ist die Bedingung in Zeile 40 nicht mehr erfüllt, und das Ende in Zeile 50 wird erreicht.

In Listing 2 wird vor dem Spielen eines Musik-Strings bereits die Weiterführung für den Fall, daß dieser bis zur letzten Note gespielt ist, in der nächsten relevanten Zeile beschlossen. In Zeile 10010 steht also an erster Stelle, in welcher Zeile (hier in Zeile 10020) die weiteren Musikbe-fehle stehen.

Die Gosub-Anweisung in Zeile 20 aktiviert den ganzen Vorgang. Bevor ein Musikeil auf seinen Schluß geprüft wird, muß dieser ja erst einmal gespielt werden.

```
[C1A] 100 ON TIMER (60) GOSUB 10000 'Routine alle 60 Se
      kunden ausführen
[FK6] 110 TIMER ON '... und zwar ab je
      tzt.
[VC6] 120 REM *** Beginn des Hauptprogrammes ***
[VG5] 130 CLS:COLOR 0,7:GOSUB 10000 'Dieses
[ZD4] 140 FOR I=1 TO 10000 'hier
[3M4] 150 LOCATE 12,37:PRINT I 'ist
[VN3] 160 NEXT I 'ein
[7Y5] 170 END 'Beispielprogramm
[T66] 9999 REM *** Ende des Hauptprogrammes ***
[876] 10000 REM *** Timerinterrupt-Routine ***
[ST6] 10010 ZEIT$=TIME$ 'sicher ist sicher.
      ..
[LT9] 10020 DEF SEG=&HB800 'Segmentadresse des
      Bildschirms bei CGA
[UY8] 10030 FOR Z=0 TO 8 STEP 2 'Zeichen für Zeiche
      n durchgehen
[ZXA] 10040 POKE Z+1,7 'Farbattribut der B
      ildschirmposition setzen
[XCB] 10050 CH=ASC(MID$(ZEIT$, (Z+2)/2,1)) 'Wert des Z
      eichens im String ermitteln
[WR5] 10060 POKE Z,CH 'Zeichen setzen
[K76] 10070 NEXT Z 'nächstes Zeichen
[T19] 10080 RETURN 'zurück zum Unterbr
      echungspunkt
```

Listing 1. Das Programm »timer.bas« zeigt die Zeit im Hintergrund an

Die Programmierung solcher Hintergrundsequenzen ist im weitesten eine Sache des Ausprobierens. Folgende Aspekte können für Sie dabei hilfreich sein:

- Sind einzelne Musikketten zu lang, wird Ihr Hauptprogramm Ablaufpausen bekommen. In einer Play-Anweisung wird nämlich immer solange verweilt, bis alle Daten gepuffert oder gespielt worden sind.
- Sind die Musikketten zu kurz, kommt es vor, daß Ihr Hauptprogramm anhält, weil der Teil in der Play-Anweisung schon bis zum Abbruchwert gespielt ist, noch bevor in das Hauptprogramm zurückgekehrt werden konnte. Somit ist der nächste Play-Interrupt fällig.
- Sind Ihre Anknüpfungswerte in der On-Play-Anweisung zu hoch, dann »reißt« die Musikkette ab, da die Abbruchbedingung unter Umständen schon erfüllt war, noch bevor sie vereinbart wurde. (db)

Wo bin ich ?

- Systemkonfiguration ermitteln

Ein Programm ist immer nur dann in der Lage, sich an die Hardware des Computers anzupassen, wenn diese ihm bekannt ist. Das gilt und funktioniert auch in Basic. Die Speicherstelle F000:FFFEhex trägt beispielsweise ein Identifikations-Byte, das unter Basic mit den Kommandos

```
def seg = &hf000:print peek(&hffe)
```

ausgelesen wird. Der Ausgabewert dieser Befehlssequenz bezeichnet den Rechnertyp wie folgt:

- PC: 255
- XT oder Tragbarer: 254
- PC jr: 253
- AT: 252

Die Adressen 0410 und 0411hex im Segment 0 beinhalten ebenfalls Informationen über den Systemaufbau. Die Bits 4 und 5 der Adresse 0410hex geben den aktuellen Bildschirmmodus an. Die Abfrage

```
print (peek(&h0410) and 48)/16
```

druckt eine Zahl zwischen eins und drei aus, für die gilt:

- 1: Farbkarte, 40 x 25
- 2: Farbkarte, 80 x 25
- 3: S/W-Karte, 80 x 25

Das Bit 0 dieser Speicherstelle gibt an, ob das System Diskettenlaufwerke besitzt, hat es den Wert 0, so führt die weitere Abfrage der Bits 6 und 7 zu einem falschen Ergeb-

nis, normalerweise aber zur exakten Anzahl der Laufwerke minus 1:

```
print (peek(&h0410) and 192)/64+1
```

Die Zahl der parallelen Schnittstellen verrät ein Blick in die letzten zwei Bits der Adresse 0411hex.

```
print (peek(&h0411) and 192)/64
```

die Zahl der seriellen Schnittstellen steht in den Bits 1, 2 und 3 dieser Speicherstelle.

```
print (peek(&h0411) and 14)/2
```

Bit 4 gibt Auskunft ob ein Gameport vorhanden ist.

```
print sgn(peek(&h0411) and 16)
```

Um sich über die Größe des Arbeitsspeichers zu informieren, fragt man die Adressen 0413hex und 0414hex ab:

```
print peek(&h0413) + peek(&h0414)*256
```

Bekommen Sie absolut unsinnige Ergebnisse, so haben Sie vergessen, vor dem Aufruf das Datensegment mit der Zuweisung »def seg = 0« auf 0 zu setzen. (db)

```
[9H1] 10 PLAY ON
[SS1] 15 GOSUB 1000
[L27] 20 CLS:FOR I=1 TO 9999:LOCATE 12,37:PRINT I:NEXT
[3T0] 30 END
[705] 1000 IF ZAEHLER=6 THEN ZAEHLER=0
[BF3] 1010 ZAEHLER=ZAEHLER+1
[EU6] 1020 ON ZAEHLER GOTO 1040,1050,1060,1070,1080,1090
[5Y5] 1040 PRINT "1":LIED$="o2mb115":GOTO 10010
[AY5] 1050 PRINT "2":LIED$="o1mb115":GOTO 10010
[KY5] 1060 PRINT "3":LIED$="o3mb115":GOTO 10010
[EY5] 1070 PRINT "4":LIED$="o2mb120":GOTO 10010
[LY5] 1080 PRINT "5":LIED$="o1mb120":GOTO 10010
[VY5] 1090 PRINT "6":LIED$="o3mb120":GOTO 10010
[319] 10010 ON PLAY(1) GOSUB 10020:PLAY LIED$+"cccc<ggg
g>":RETURN
[H49] 10020 ON PLAY(1) GOSUB 10030:PLAY "cccc<gg>c<baaa
aaaa":RETURN
[C69] 10030 ON PLAY(1) GOSUB 10040:PLAY "aaaaaeagfffc
ccc":RETURN
[799] 10040 ON PLAY(1) GOSUB 10050:PLAY "fffcdefggggddd
dggg":RETURN
[P59] 10050 ON PLAY(1) GOSUB 10060:PLAY "gagab<cccc<ggg
g>cc":RETURN
[659] 10060 ON PLAY(1) GOSUB 10070:PLAY "cdedc<baaaaae
aaab":RETURN
[C79] 10070 ON PLAY(1) GOSUB 10080:PLAY ">c<bagffffcccc
fff":RETURN
[YS9] 10080 ON PLAY(1) GOSUB 1000:PLAY "cdefggggdddg
ggagab>":RETURN
```

Listing 2. Im Hintergrund Musik spielen lassen mit »play.bas«

Lieben Sie es...

abends über Ihrer Buchführung zu brüten?

Lästige Routinearbeiten nimmt Ihnen unsere professionelle Büro-Software ab:

Auftrag-Angebot-Lager-Faktura

Angebot, Auftrag, Rechnung, Lieferschein, Lager, Mahnung, Bestellwesen, Kunden, Lieferanten, Serienbriefe usw.

499.-

Finanzbuchhaltung

Bilanz, SS, BWA, Journal, DATEV-Kontenrahmen, GuV, Ust.-Vorabm., Kreditoren, Debitoren usw.

499.-

Lohnbuchhaltung

KK- und Finanzamtliste, Lohnabrechnung, Überweisungsträger usw.

499.-

Branchenlösungen

- Bauhaupt- und Nebengewerbe, Architekten
- Versicherungsagenturen ■ Autovermietungen

Preis auf Anfrage

Alle Programme

- ▲ extrem einfach in der Bedienung
- ▲ menügesteuert mit Fenstertechnik
- ▲ individuell anpaßbar, mehrplatzfähig
- ▲ Ihr Netzwerk-Spezialist

Wir schicken Ihnen gern kostenlos eine Demo-Disk und informieren Sie unverbindlich über unsere Büro-Software: ► **SOFTWARE & HARDWARE, Ingenieurbüro Bitter**, Telefon: 0421/210511 Hauptmann-Böse-Weg 2-4, D-2800 Bremen 1 ► **DELA-EDV**, Köln 1, Telefon: 0221/517084 ► **Büttner-EDV**, Bielefeld, Telefon: 0521/493994

Der Boot-Record

Kaum ein PC-Benutzer weiß, was passiert, wenn sein Computer das Betriebssystem von der Diskette liest. Dabei sind die beim Bootvorgang ablaufenden Vorgänge gar nicht so kompliziert, wie man allgemein glaubt.

Jeder Benutzer eines PC hat sich sicher schon einmal gefragt, was in seinem Computer geschieht, wenn er ihn einschaltet. Denn bevor Sie Ihren Computer benutzen können, benötigt der Computer einen relativ langen Zeitraum, ehe er sich mit der Eingabeaufforderung (dem »Prompt«) bereitmeldet. Dieser Vorgang, der die Hardware des Computers testet und das Betriebssystem lädt, nennt man »Initialisierung« oder »Booten«.

Die Initialisierung läßt sich in zwei Phasen unterteilen: Bevor das angeschlossene Disketten- oder Festplattenlaufwerk angesprochen wird, initialisiert der PC die Hardware und daraufhin das Betriebssystem des Computers.

Die Routine für die Hardware-Initialisierung und der Selbsttest werden an der Adresse FFFF0hex gestartet, dem sogenannten »CPU Reset Entry Point« (der Startadresse der CPU bei einem Zurücksetzen des PC). Der in dieser Adresse gespeicherte Wert gibt den Anfang der Routine an, die alle notwendigen Aufgaben erfüllt. Sie initialisiert die Hardware, führt Selbsttests durch und formatiert den Variablen- und Parameterbereich des »POST« (»Power On Self Test«). Der »POST« benutzt den Arbeitsspeicher des PC erst, wenn dieser erfolgreich überprüft worden ist.

Die Initialisierung läuft zusammengefaßt in folgender Weise ab:

- Verhindern der »NMI« (»Nicht Maskierbaren Interrupts«)
- POST-Selbsttest

- NVR-Test (batteriegepufferter Systemspeicher)
- Initialisierung verschiedener Hardwarebausteine
- Initialisierung des POST-Variablenbereichs
- Initialisierung der POST-Interrupts
- Initialisierung der VDU
- Initialisierung des 8259 Interrupt Controllers
- Ausgabe der Einschaltmeldung
- NMI erlauben
- Initialisierung aller externen ROMs
- Booten des MS-DOS

Ein Problem ergibt sich beim letzten Punkt. Wie soll man auf unterschiedlichen Computern dieselbe Boot-Routine verwenden, wenn doch die Hardware-Voraussetzungen völlig unterschiedlich sind? Die Entwickler von Microsoft und IBM fanden eine geniale Lösung: den sogenannten Boot-Record.

Die Idee liegt darin, die Informationen darüber, wie der Computer geladen werden muß, auf der Diskette statt im ROM abzulegen. Bei einem neuen und weiterentwickelten PC genügt es dann, den Boot-Record zu ändern, um die Boot-Routine an die gesteigerte Leistungsfähigkeit anzupassen.

Der Boot-Record ist eine der bedeutendsten Parameterstrukturen des MS-DOS-Betriebssystems. Er enthält neben den wichtigsten Daten des angeschlossenen Festplatten- oder Diskettenlaufwerks eine Routine zum Booten.

Mit den Informationen des Boot-Records kann man nun einerseits ein Betriebssystem (oder ein anderes Programm) laden und andererseits das Laufwerk beim System anmelden. Der Boot-Record befindet sich immer im logischen Sektor 0 des Disketten- oder Festplattenlaufwerkes. Die darin enthaltenen Informationen können Sie der Tabelle entnehmen.

An diese Parametertabelle hängt man nun eine Routine zum Booten des Computers an. Auf diese Routine beziehungsweise deren Einsprungadresse weisen die ersten drei Byte des Boot-Records.

Das Boot-ROM reserviert zunächst einen 512 Byte großen Speicherpuffer, der bei der Adresse 0000:7C00hex beginnt und bei 0000:7DFFhex endet. Danach wird versucht, höchstens viermal das angeschlossene Disket-

tenlaufwerk zurückzusetzen. Ist dieser Vorgang erfolgreich, dann benutzt die ROM-Routine den Interrupt 13hex, um den ersten Sektor von Seite Null, Spur Null auf Laufwerk A in den eingerichteten Puffer zu laden. Ist das erste Wort (entspricht zwei Byte) in dem Puffer gleich Null oder gleicht es den folgenden zehn Byte, dann gibt die Routine eine Fehlermeldung aus und hält an. Ist hingegen der Leseversuch erfolgreich, so führt die Routine danach einen Sprung zu der Anfangsadresse des Puffers beziehungsweise den Beginn des Boot-Records durch.

Hier befinden sich die drei Byte, die die Boot-Routine des Boot-Records aufrufen. Die genaue Form dieses Sprunges ist je nach verwendetem Computer unterschiedlich. Es kann sich dabei sowohl um einen zwei Byte langen »Short-jump« als auch um einen drei Byte langen »Far-jump« handeln.

Tritt nun aber der Fall ein, daß das Diskettenlaufwerk nicht bereit ist, weil beispielsweise keine Diskette eingelegt, die Verschlussklappe noch geöffnet oder gar kein Diskettenlaufwerk angeschlossen ist, greift das Boot-ROM bis zu dreimal auf die Festplatte zu, um den logischen ersten Sektor von der Festplatte in den Puffer zu lesen. Funktioniert dies, vergleicht es das zuletzt gelesene Wort mit dem Wert AA55hex. Sind beide identisch, dann folgt wiederum der Sprung zu der ersten Anweisung im Puffer. In allen anderen Fällen wiederholt sich dieser Vorgang. War dagegen ein Booten von der Festplatte nicht erfolgreich, so springt das ROM den Interrupt 18hex an. Bei älteren Computern befindet sich hier der Sprung ins ROM-Basic, die neueren PCs geben die Meldung »Bitte eine Systemdiskette einlegen« aus. Bei Festplatten kann der Boot-Record an einer anderen Stelle als am physikalischen Anfang des Datenträgers liegen, was beispielsweise bei unterschiedlichen Partitionen der Festplatte der Fall ist. Wichtig ist nur, daß es logisch der Sektor Null des MS-DOS-Bereiches einer Diskette oder Festplatte ist.

Die Funktionsweise dieses Boot-ROMs und des BOOT-Sektors zu verstehen, ist wesentlich für das Verständnis des gesamten Betriebssystems MS-DOS. Wenn Sie die interne Struktur dieses wichtigen Systemvorgangs erkannt haben, ist es auch nicht mehr schwer, ihn zu verändern. So verstehen Sie dann beispielsweise auch die internen Systemroutinen problemlos, wenn Sie diese Mithilfe von »Debug« oder einem Programm wie dem »BCI Disasm« disassemblieren.

(Martin Althaus/Markus Zietlow/gh)

Tabelle. Der Inhalt des Boot-Records, der sich immer im logischen Sektor Null des Diskettenlaufwerks oder der Festplatte befindet

Byte	Bedeutung
00h-02h	Sprungbefehl zum Boot-Programm
03h-0Ah	Name der Diskette (8 Bytes ASCII)
0Bh-0Ch	Anzahl der Bytes pro Sektor
0Dh	Anzahl der Sektoren pro Cluster
0Eh-0Fh	Anzahl der Sektoren des Boot-Records
10h	Anzahl der Kopien des FAT
11h-12h	Anzahl der Einträge im Stammverzeichnis
13h-14h	Anzahl der Sektoren der Diskette/Platte
15h	Identifizierungscode der Diskette/Platte
16h-17h	Anzahl der Sektoren des FAT
18h-19h	Anzahl der Sektoren pro Spur
1Ah-1Bh	Anzahl der Diskettenseiten/Plattenköpfe
1Ch-1Dh	Anzahl der reservierten Sektoren

16 Farben für die CGA-Karte

Bereits im April dieses Jahres stellte Ihnen die DOS International ein Verfahren vor, mit dem sich auf der CGA-Farbgrafikkarte neben den Auflösungen von 640x200 und 320x200 Bildpunkten noch eine dritte Auflösung mit 160x100 Bildpunkten in gleichzeitig 16 Farben darstellen läßt. Nun liegt eine verbesserte Lösung vor, die die rechnerische Auflösung von 160x100 Bildpunkten voll unterstützt.

Die Grafikauflösungen der CGA-Farbgrafikkarte mit 320x200 Bildpunkten in vier und 640x200 Bildpunkten in zwei Farben sind bestens bekannt. Kaum bekannt ist hingegen die Tatsache, daß der Videocontroller 6845 der CGA-Farbgrafikkarte auch einen dritten Grafikmodus mit 160x100 Bildpunkten in 16 Farben erlaubt. Über diesen 16-Farben-Modus gibt es so gut wie keine Informationen, selbst das technische Handbuch für den IBM PC/XT spricht lediglich von einer »speziellen Programmierung« des Videocontrollers.

Der 16-Farben-Modus ist genaugenommen kein richtiger Grafikmodus, sondern eher ein verkappter Textmodus, denn für diesen Modus wird der Videocontroller der CGA-Farbgrafikkarte so umprogrammiert, daß alle vier Bildschirmseiten des farbigen 80-Zeichen-Textmodus gleichzeitig angezeigt werden. Dadurch sind jeweils nur die obersten zwei Bildzeilen eines Zeichens sichtbar, so daß die Position eines Zeichens jeweils von vier untereinanderliegenden Doppelbildzeilen gebildet wird.

Wie im gewöhnlichen Textmodus kann auch in diesem Fall jede Zeichenposition 16 verschiedene Farben annehmen. Genauer gesagt läßt sich die Schriftfarbe aus 16 Farben und zusätzlich die Hintergrundfarbe aus acht Farben wählen. Auch kann für jede Zeichenposition die Blinkfunktion aktiviert werden. Diese Farb- und Blinkinformationen zu einer Zeichenposition sind in einem Attributbyte gespeichert, das im Bildschirm-

speicher direkt auf den ASCII-Wert des zugehörigen Zeichens folgt.

Der Bildschirmspeicher der CGA-Farbgrafikkarte beginnt ab der Adresse B8000hex und ist im Textmodus in der Reihenfolge

Code für das erste Zeichen
Attribut für das erste Zeichen
Code für das zweite Zeichen
Attribut für das zweite Zeichen etc.

aufgebaut. Den Aufbau des Attributbyte zeigt Tabelle 1, und Tabelle 2 verdeutlicht, wie sich aus den Bits des Attributbyte 16 verschiedene Farben zusammensetzen lassen.

Die Funktion von Bit 7 des Attributbyte ist von Bit 5 des Modusauswahlregisters (Portadresse 3D8hex) auf der CGA-Farbgrafikkarte abhängig. Wenn dieses Bit gesetzt ist, was nach der Initialisierung der CGA-Karte durch das BIOS immer der Fall ist, läßt ein gesetztes Bit 7 im Attributbyte die zugehörige Zeichenposition blinken. Ist das Bit 5 des Modusauswahlregisters dagegen gelöscht, so bewirkt das gesetzte Bit 7 des Attributbyte eine intensive Hintergrundfarbe. Auf diese Weise stehen Ihnen dann alle 16 Farben sowohl als Vordergrund- wie auch als Hintergrundfarbe zur Verfügung.

Weil das Modusauswahlregister nur beschrieben, nicht aber gelesen werden kann, von den übrigen Bits dieses Registers aber auch der Videomodus abhängt, müssen Sie immer den aktuellen Modus kennen, wenn Sie Bit 5 löschen wollen, um die restlichen Bits nicht versehentlich zu ändern.

Über das Modusauswahlregister der CGA-Farbgrafikkarte läßt sich auch der aktuelle Videomodus definieren

So entspricht der 16-Farben-Modus dem farbigen 80-Zeichen-Textmodus, für den Bit 0 des Modusauswahlregisters gesetzt sein muß. Bit 1 muß auf 0 gesetzt sein, damit der Textmodus ausgewählt ist, andernfalls befindet sich die CGA-Farbgrafikkarte im Grafikmodus. Bit 2 bestimmt, ob die Bildschirmausgabe farbig (Bit 2 = 0) oder schwarzweiß (Bit 2 = 1) erfolgt. Von Bit 3 hängt es ab, ob die Bildschirmausgabe aktiv oder inaktiv ist. Wenn Sie dieses Bit löschen, wird die Bildschirmausgabe (beispielsweise während der Neuprogrammierung des Videocontrollers 6845) unterdrückt. Der Zustand von Bit 4 ist im Textmodus nicht relevant und Bit 6 sowie Bit 7 werden überhaupt nicht benutzt.

Mit dem Basic-Befehl

```
out &H3d8,9
```

setzen Sie folglich Bit 5 und lassen die Bildschirmanzeige eingeschaltet, während Sie mit

```
out &H3d8,1
```

die Bildschirmanzeige wieder abschalten.

Der Aufbau des Attributbyte

Bit	Bedeutung
0	Vordergrund blau
1	Vordergrund grün
2	Vordergrund rot
3	Vordergrund intensiv
4	Hintergrund blau
5	Hintergrund grün
6	Hintergrund rot
7	Hintergrund blinkend/intensiv

Tabelle 1. Der Inhalt des Attributbyte bestimmt die Farbe des Vorder- und Hintergrundes

Die 16 Farben der CGA-Farbgrafikkarte

Intensitäts-Bit	Rot-Bit	Grün-Bit	Blau-Bit	Farbe	Wert
0	0	0	0	Schwarz	0
0	0	0	1	Blau	1
0	0	1	0	Grün	2
0	0	1	1	Kobaltblau	3
0	1	0	0	Rot	4
0	1	0	1	Violett	5
0	1	1	0	Braun	6
0	1	1	1	Weiß	7
1	0	0	0	Grau	8
1	0	0	1	Hellblau	9
1	0	1	0	Hellgrün	10
1	0	1	1	Hellkobaltblau	11
1	1	0	0	Hellrot	12
1	1	0	1	Hellviolett	13
1	1	1	0	Gelb	14
1	1	1	1	Hellweiß	15

Tabelle 2. Mit den drei Farbsignalen und dem Intensitätssignal der CGA-Karte lassen sich 16 verschiedene Farben erzeugen

Um den Videocontroller 6845 zu veranlassen, alle vier Textseiten gleichzeitig auf dem Bildschirm anzuzeigen, müssen Sie die ersten zehn Register dieses Bausteins mit neuen Werten beschreiben. Tabelle 3 listet die Werte für diese zehn Register auf. Bei einem Schreibzugriff auf die Register des Videocontrollers müssen Sie zunächst über das Adreßregister (Portadresse 3D4hex) das gewünschte interne Register auswählen, und dann in das Datenregister (Portadresse 3D5hex) den für das interne Register bestimmten Wert schreiben. Weil während der Neuprogrammierung des Videocontrollers unkontrollierte Signale zum Monitor gelangen, die diesen unter Umständen beschädigen können, läßt sich die Bildschirmausgabe während der Programmierung des Controllers unterdrücken, indem das Bit 3 des Modusauswahlregisters gelöscht wird. In Basic dient dazu der Befehl

```
out &H3d8,1
```

Erst danach laden Sie die Register des Videocontrollers mit den neuen Werten und schalten abschließend mit

```
out &H3D8,9
```

die Bildschirmausgabe wieder ein. Im 16-Farben-Modus läßt sich jetzt die Hintergrundfarbe über die vier oberen Bit des Attributbyte beliebig setzen, sofern alle vier Bildschirmseiten leer sind. Bei 80 Zeichen pro Zeile und 4x25 Bildschirmzeilen ergibt sich so eine Auflösung von 80x100 Punkten, weil die Farbe jeder Zeichenposition von der Hintergrundfarbe bestimmt wird, solange kein Zeichen an dieser Position angezeigt wird (beziehungsweise nur Zeichen, deren zwei obersten Zeilen leer sind).

Über einen Trick läßt sich die Bildpunktauflösung im 16-Farben-Modus verdoppeln

Niemand zwingt Sie aber, die Zeichenposition leer zu lassen, das heißt, nur die ASCII-Zeichen 0, 32 oder 255 zu benutzen. Wenn Sie beispielsweise an einer Zeichenposition

das Zeichen mit dem ASCII-Wert 222 ausgeben, ist die rechte Hälfte der Zeichenposition ausgefüllt und die linke leer, mit anderen Worten: Die Farbe der linken Hälfte des Zeichens wird durch die vier oberen Bit (Hintergrundfarbe) des Attributbyte und die rechte Hälfte durch die vier unteren Bit (Vordergrundfarbe) des Attributbyte bestimmt. So einfach läßt sich die horizontale Auflösung verdoppeln!

Der 16-Farben-Modus ist am einfachsten zu programmieren, wenn Sie nach der Initialisierung des Videocontrollers 6845 mit den Werten aus Tabelle 3 an alle Zeichenpositionen auf allen vier Bildschirmseiten das ASCII-Zeichen 222 ausgeben. Danach läßt sich allein über das Attributbyte die Farbe jedes Bildpunktes manipulieren. Gleichzeitig ist es sinnvoll, alle Attributbytes auf den Wert Null (für einen schwarzen Vorder- und Hintergrund) zu setzen, so daß erst einmal alle Bildpunkte gelöscht sind. Wie bereits erwähnt, beginnt der Bildschirmspeicher der CGA-Farbgrafikkarte ab der Adresse B8000hex. Wenn Sie das Datensegment auf den Wert B000hex setzen, errechnet sich der Offset eines Bildpunktes im 16-Farben-Modus nach folgender Formel:

$$\text{offset} = 160 \times y + 2 \times [\text{int}(x/2)] + 1$$

Wenn die Variable x gerade ist, so ist die untere Hälfte (Bit 0 bis Bit 3) des Attributbyte für die Farbe des Bildpunktes zuständig, und ist der Wert von x gerade, so gibt die obere Hälfte (Bit 4 bis Bit 7) des Attributbyte die Farbe des Bildpunktes an. Achten Sie beim Zugriff auf einen Bildpunkt darauf, nur die dafür zuständigen Bits zu verändern.

Listing 1 zeigt die Programmierung des 16-Farben-Modus für die CGA-Farbgrafikkarte in GW-Basic. Es besteht aus einem kleinen Demonstrationsprogramm von Zeile 20 bis Zeile 940 und sechs Unterprogrammen, die ab den Programmzeilen 1000, 2000, 3000, 4000, 5000 und 6000 beginnen. Die Routinen ersetzen zum Teil Basic-Befehle wie beispielsweise »pset« (Setzen eines Bildpunktes) oder »print«, die im 16-Farben-Modus natürlich nicht funktionieren.

Ein GW-Basic-Programm zeigt, wie Sie den 16-Farben-Modus der CGA-Farbgrafikkarte programmieren

Die Initialisierungsroutine ab Zeile 1000 schaltet den farbigen 80-Zeichen-Textmodus mit

```
screen 0,1
```

ein, unterdrückt die Bildschirmanzeige und verändert die Bedeutung des Blinkbit mit

```
out &H3d8,1
```

Darauf lädt die Routine den Videocontroller mit den für den 16-Farben-Modus erforderlichen neuen Werten und kopiert eine Assembleroutine aus den beiden Data-Zeilen 1050 und 1060 in den mit

```
clear,65507
```

reservierten Speicherbereich. Die Assembleroutine (Listing 2) schreibt an jede Zeichenposition im Bildschirmspeicher der CGA-Farbgrafikkarte das Grafikzeichen mit dem ASCII-Wert 222 und setzt die Vorder- und Hintergrundfarbe in allen Attributbytes durch den Wert Null auf schwarz.

Die Assembleroutine ist nötig, weil immerhin 16000 Adressen beschrieben werden müssen und eine Basic-Routine aus Zeitgründen dazu nicht ausreicht. Bevor die Assembleroutine jedoch in den Speicher kopiert werden darf, muß zuvor Speicherplatz für sie reserviert werden. Dies geschieht mit dem Clear-Befehl

```
clear,65507
```

Dieser Befehl steht zweckmäßigerweise am Anfang des Programms, weil er zugleich den Inhalt aller Variablen löscht.

Nach dem Aufruf der Assembleroutine schaltet das Basic-Programm die Bildschirmanzeige mit

```
out &H3D8,9
```

wieder ein.

Weil Turbo Basic den ihm zur Verfügung stehenden Speicher anders organisiert als GW-Basic, müssen Sie bei Verwendung dieses Compilers den Maschinencode aus den Data-Zeilen 1050 und 1060 in eine Inline-Prozedur einbinden. Dann ist allerdings der letzte Wert 203, der für den Assemblerbefehl »retf« steht, ersatzlos zu streichen. Turbo Basic fügt nämlich den durch »inline« angegebenen Maschinencode direkt in den kompilierten Programmcode ein. Mit der zweiten Routine ab Zeile 2000 in Listing 1 läßt sich die Randfarbe beliebig setzen. Dazu weisen Sie der Variable RF% vor dem Aufruf einen Wert zwischen 0 und 15 zu. Ab Zeile 3000 steht die Routine zum Setzen eines Bildpunktes. Mit den Variablen X% und Y% übergeben Sie die Koordinaten und mit farbe% die Farbe des zu setzenden Bildpunktes. Mit

```
gosub 4000
```

kehren Sie in den Textmodus zurück. Eine recht nützliche Routine beginnt ab Zeile 5000. Unter Verwendung der Assembleroutine des Initialisierungsteils löscht sie den Bildschirm und setzt die Farbe aller Punkte auf den Wert, den Sie in der Variable HF% übergeben. Dabei modifiziert das Programm die ursprüngliche Assembleroutine während des Programmablaufs.

Register	Wert	Register	Wert
0	113	5	6
1	80	6	100
2	90	7	112
3	10	8	0
4	127	9	1

Tabelle 3. Diese Werte müssen Sie in die ersten zehn internen Register des Videocontrollers 6845 schreiben, um den 16-Farben-Modus zu initialisieren

Die letzte Routine ab Zeile 6000 erlaubt die Textausgabe im 16-Farben-Modus. Dabei macht sie sich die Tatsache zunutze, daß bei einem gewöhnlichen BIOS ab der Adresse FFA6Ehex eine Tabelle beginnt, die die Bildpunktmatrix der ASCII-Zeichen 0 bis 127 im Grafikmodus definiert. Während im Textmodus das Aussehen der Zeichen von dem auf der CGA-Farbgrafikkarte enthaltenen Zeichensatz-EPROM beziehungsweise -ROM festgelegt wird, holt sich das BIOS im Grafikmodus die Punktdaten der Zeichen aus der erwähnten Tabelle.

Im 16-Farben-Modus lassen sich mit etwas programmtechnischem Geschick auch Textzeichen ausgeben

Das Aussehen der Zeichen mit ASCII-Werten über 127 legt eine zweite Tabelle fest, auf die der Interruptvektor des Interrupt 1Fhex zeigt. Diese Tabelle der Anwender erst anlegen, beispielsweise durch den MS-DOS-Befehl

»graphtabl«. Darauf ist es auch zurückzuführen, daß im Grafikmodus anstelle von Umlauten und anderen Zeichen mit einem ASCII-Wert größer als 127 nur Unsinn auf dem Bildschirm erscheint, sofern Sie vergessen haben, »graphtabl« zuvor zu installieren.

Die Routine in Listing 1 schreibt allerdings nur Zeichen mit den ASCII-Werten von 0 bis 127 auf den Bildschirm; alle anderen Zeichen werden ignoriert. In der Zeichenkette »T\$« übergeben Sie den zu schreibenden Text, in den Variablen X1% und Y1% die Koordinaten der Startposition und in der Variablen farbe% die Schriftfarbe.

Natürlich ist der 16-Farben-Modus der CGA-Farbgrafikkarte mit Einschränkungen verbunden:

- Weil das Programm direkt in die Register des Videocontrollers 6845 schreibt, läuft der 16-Farben-Modus nur mit einer echten CGA-Karte oder mit EGA-Karten, die bis hinab auf die Registerebene CGA-kompatibel sind. Dies trifft bei den meisten EGA-Karten nicht zu.

– Bei einfachen CGA-Karten erscheint störender »Schnee« auf dem Bildschirm, wenn die Routine auf den Bildschirmspeicher der CGA-Karte zugreift. Der versierte Assemblerprogrammierer umgeht dieses Manko, indem er die Routine zum Setzen eines Punktes in Assemblersprache schreibt und die Bildschirmausgabe über Bit 3 des Moduswahlregisters während der Lese- oder Schreibzugriffe auf den Bildschirmspeicher abschaltet.

Es gibt im Bereich der Public-Domain-Programme auch einige Spiele, die den 16-Farben-Modus der CGA-Farbgrafikkarte benutzen, wie zum Beispiel »bricks.exe« von Vincent Bly. Dabei handelt es sich um eine recht gute Version von »Breakout«. Sicherlich fallen Ihnen noch Erweiterungen zu den hier vorgestellten Routinen ein, beispielsweise für Grafikfunktionen, die eine Linie oder einen Kreis zeichnen. Wir wünschen Ihnen viel Erfolg und sind auf Ihre Ergebnisse gespannt.

(Martin Hepp/Tilmar Kümmell/ma)

★ **MultiScreen ist keine Hexerei:**

★ **DIE NEUE BILDSCHIRMDIMENSION**

★ für Geschäftsgrafiken,
★ Zeichenprogramme und
★ Spielprogramme aller Art.

★ Sie kennen das Problem:

★ Auf Ihrem PC mit Monochrom-Monitor und Herculesgrafikkarte funktionieren einfach keine Grafikspiele. Dieselben Schwierigkeiten erleben Sie bei Geschäftsgrafiken, Zeichenprogrammen und CAD-Anwendungen. Der Bildschirm bleibt dunkel.

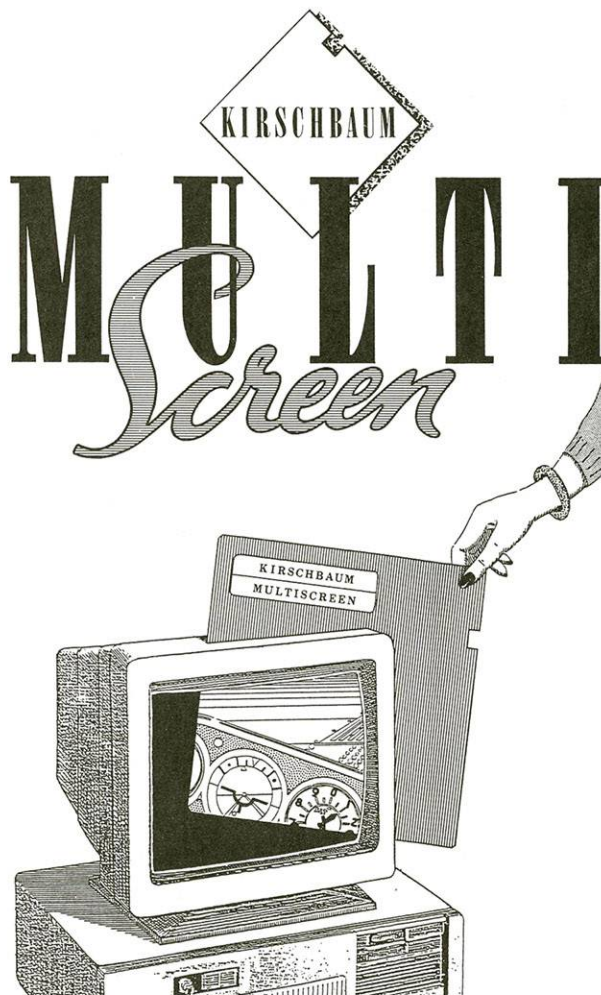
★ Der Grund: Die meisten Grafikprogramme unterstützen nicht den Hercules-Standard. Selbst das auch beinahe in allen PC's verfügbare GW-BASIC zaubert keine Grafik auf Ihren Bildschirm. Ist es Ihnen nicht auch schon so ergangen? Sie geben immer wieder die Grafikbefehle LINE, CIRCLE, DRAW, etc. ein und sind dabei fast verzweifelt ...

★ Schluß damit! Kein dunkler Bildschirm mehr! Wir haben das Problem gelöst!

★ MultiScreen II Version 3.3 DM 112,-
(für IBM und Kompatible)

★ MultiScreen 1640 Version 2.2 DM 98,-
(nur für AMSTRAD PC 1640 MD)

★ MultiScreen EURO Version 1.1 DM 98,-
(nur für Schneider EURO-PC)
★ (unverbindliche Preisempfehlung)



... und das bietet Ihnen MultiScreen:

■ MultiScreen emuliert auf Ihrem IBM XT/AT (und alle Kompatiblen wie z.B. Commodore PC III oder TANDON) mit HERCULES-Monochrom Grafikkarte (oder kompatibler Grafikkarte) folgende Grafikauflösungen:

320 × 200 Punkte 3 Graustufen CGA kompatibel
640 × 200 Punkte monochrom CGA kompatibel
640 × 400 Punkte monochrom Olivetti kompatibel
Herculesgrafik mit 720 × 348 Punkte ohne Einschränkung.

■ Alle BIOS-Aufrufe werden für diese Modi unterstützt.
■ CGA-Programme werden ohne Geschwindigkeitsverlust emuliert.

■ Spielprogramme werden Wirklichkeit: DEFENDER OF THE CROWN, GATO, PITSTOP II, PC-GOLF, SARGON III, GUNSHIP, KING'S QUEST, DIGDUG, BUSHIDO, JET, NEWSROOM, KEYDRAW, PC-PG, THE DRAFTSMAN, PACMAN, STRIKER, LODERUNNER, etc. ...

■ Verzichten Sie nicht länger auf Grafik bei Programmen wie GWBASIC, TURBO PASCAL, PRINTMASTER, GRAFIK ASSISTANT, etc.

■ Einfache Handhabung bei selbststartenden Programmen. Dadurch keine Probleme bei Spielen wie TAPPER, SUMMER GAMES, WINTER GAMES, CALIFORNIA GAMES, WORLD GAMES, DECATHLON, NIGHTMISSION PINBALL, FLIGHTSIMULATOR von Microsoft.

■ Alle Programme, die bei der Umschaltung von Text auf Grafik und umgekehrt über das BIOS laufen (GWBASIC, BASICA, TURBO PASCAL, Printmaster, PC-PAINT, Framework, Enable, usw.) erscheinen absolut ohne Probleme.

■ Für Programme, bei denen die Umschaltung von Text auf Grafik nicht über das BIOS geht, bietet MultiScreen eine Hot-Key-Taste. Dadurch ist selbst der Flugsimulator ohne Einschränkung lauffähig.

■ Software ohne Probleme auf Festplatte installierbar.

■ MultiScreen ist voll in Assembler geschrieben und bleibt resident im RAM-Speicher, verbraucht jedoch nur 9 KB.

■ Auslieferung mit ausführlicher deutscher Benutzeranleitung.

KIRSCHBAUM SOFTWARE GmbH

Kronau 15, D-8091 Emmering,
Tel. 08067/1220 oder 1051,
Fax. 08067/1053

Schweiz: DSE Datensystems-Engineering AG, Badener Str. 262,
CH 8004 Zürich.

Österreich: Überreuter Media Handels- u. Verlagsges. m.b.H.,
Laudongasse 29, A-1082 Wien.

Wir suchen noch weitere Händler!


```
[U12] 10 CLEAR ,655071
[84B] 20 REM Demonstration des 160x100-Punkte-Modus der
      CGA-Farbgrafikkarte
[XM8] 30 REM (C) 1988 by DOS International, Autor: Mart
      in Hepp
[CZB] 40 CLS:PRINT "Bitte drücken Sie eine Taste, um da
      s Programm zu starten."
[Z04] 50 IF INKEY$="" THEN GOTO 50
[0V7] 60 GOSUB 1000 'Initialisierungsroutine aufrufen
[524] 70 LET RF%=5 'Randfarbe=violett
[BU4] 80 GOSUB 2000 'Randfarbe setzen
[42A] 85 HF%=11:GOSUB 5000 'Hintergrundfarbe auf Helkob
      altblau setzen
[NZ1] 90 FOR D=0 TO 15
[1X2] 100 LET farbe%=15-D
[W77] 110 LET Y%=D:FOR X%=D TO 158-D:GOSUB 3000:NEXT X%
[0W7] 120 FOR Y%=D TO 98-D+1:GOSUB 3000:NEXT Y%:LET Y%=
      Y%-1
[BX8] 130 FOR X%=158-D TO D STEP -1:GOSUB 3000:NEXT X%:
      LET X%=X%+1
[317] 140 FOR Y%=98-D TO D STEP -1:GOSUB 3000:NEXT Y%
[CG1] 150 NEXT D
[NH7] 160 farbe%=12:X1%=17:Y1%=20:T$="16 Farben":GOSUB
      6000
[SH8] 170 farbe%=14:X1%=17:Y1%=30:T$="by Martin Hepp":G
      OSUB 6000
[0C9] 180 farbe%=13:X1%=17:Y1%=40:T$="DOS International
      ":GOSUB 6000
[D74] 910 IF INKEY$="" THEN GOTO 910
[216] 930 GOSUB 4000 '16-Farben-Modus beenden
[BD3] 940 END 'Programmende
[RD7] 1000 'Initialisierung des 160x100-Punkte-Modus
[VM0] 1010 '
[527] 1020 KEY OFF: SCREEN 0,1:CLS:OUT &H3D8,1:RESTORE
      1040
[CR8] 1030 FOR A%=0 TO 9: READ B%:OUT &H3D4,A%:OUT &H3D
      5,B%:NEXT A%
[YBB] 1040 DATA 113,80,90,10,127,6,100,112,0,1:REM
      Neue
      Werte für Videocontroller
[U5A] 1050 DATA 30,184,0,184,142,216,191,0,0:REM Daten
      für Assemblerroutine
[S38] 1060 DATA 185,64,31,176,222,136,5,71,176,0,136,5,
      71,226,244,31,203
[4VA] 1070 DEF SEG:RESTORE 1050:FOR C%=0 TO 25:READ W%:
      POKE 65508!+C%,W%:NEXT C%
[LYA] 1080 SUBRT=65508!:CALL SUBRT '16-Farben-Bi
      ldschirm initialisieren
[13A] 1090 OUT &H3D8,9 'Bildschirm
      zeige wieder einschalten
[VU1] 1100 RETURN
[YM0] 1110 '
[MG3] 2000 'Randfarbe setzen
[E25] 2010 'RF% enthält die Randfarbe
[1N0] 2020 '
[ZDA] 2030 IF RF%<0 OR RF%>15 THEN RETURN 'unzulässige
      Werte abfangen
[CCA] 2040 OUT &H3D9,RF% 'Wert in das
      »Output Color Register«
[0R4] 2050 'schreiben
[KV1] 2060 RETURN
[NN0] 2070 '
[GX2] 3000 'Punkt setzen
[TT4] 3010 'X% enthält X-Koordinate
[4U4] 3020 'Y% enthält Y-Koordinate
[R54] 3030 'farbe% enthält Farbe
[UAE] 3040 IF X%<0 OR X%>159 OR Y%<0 OR Y%>99 OR farbe%
      <0 OR farbe%>15 THEN RETURN 'ungültige Werte
[Y2C] 3050 DEF SEG=&HB800 'Peek und Poke beziehen sich
      auf den Bildschirmspeicher
[6NA] 3060 LET ADR%=160*Y%+2*(INT (X%/2))+1 'Adresse de
      s Bildpunktes berechnen
[WAP] 3070 IF (X% MOD 2)=0 THEN WERT%=(PEEK(ADR%) MOD
      16) + farbe%*16 ELSE WERT%=(INT(PEEK(ADR%)/16)*1
      6+ farbe%) 'Neuen Wert für diese Adresse berechne
      nn
[YZ2] 3080 POKE ADR%,WERT%
[0W1] 3090 RETURN
[YM0] 3100 '
[YG6] 4000 'Zurücksetzen des Bildschirmmodus
[4B2] 4010 SCREEN 0,0,0
[6V1] 4020 RETURN
[9N0] 4030 '
[BP6] 5000 '16-Farben-Modus-Bildschirm löschen
[3PB] 5010 'HF% enthält die Farbe, die alle Punkte anne
      hmen (Hintergrundfarbe)
[152] 5020 '0<= HF% < 16
[BN0] 5030 '
[MR1] 5040 DEF SEG
[ZTU] 5050 IF PEEK (65508!)<> 30 THEN BEEP:SCREEN 0,0,0
      :PRINT "Vor dem Aufruf dieser Routine muß die Ass
      embleroutine in den Speicher kopiert worden sein
      ":BEEP:BEEP:STOP
```

```
[ZCA] 5060 IF HF%<0 OR HF%>15 THEN RETURN 'unzulässige
      Werte abfangen
[177] 5070 POKE 65526!,HF%+16*HF% 'CLS-Routine patchen
[N72] 5080 OUT &H3D8,1
[8L6] 5090 SUBRT=65508!:CALL SUBRT 'und aufrufen
[072] 5100 OUT &H3D8,9
[7V1] 5110 RETURN
[9V5] 6000 'Text im 16-Farben-Modus ausgeben
[ZN4] 6010 'übergebene Parameter:
[61B] 6020 'T$ enthält den auszugebenden Text (nur ASCII
      I-Zeichen 0 bis 127)
[C47] 6030 'X1% = X-Koordinate der Anfangsposition
[P47] 6040 'Y1% = Y-Koordinate der Anfangsposition
[TT9] 6050 'farbe% = Schriftfarbe, in der der Text ausg
      egeben wird
[TN0] 6060 '
[8P2] 6070 DEF SEG=&HF000
[US6] 6080 FOR J%=0 TO LEN(T$)-1:U%=ASC(MID$(T$,J%+1,1)
      )
[F44] 6090 IF U%>127 THEN GOTO 6160
[GY5] 6100 FOR Z%=0 TO 7:K%=PEEK(&HFA6E+Z%*8*U%)
[9L3] 6120 FOR O%=7 TO 0 STEP -1
[CP9] 6130 IF (K% AND 2^O%)=2^O% THEN X%=X1%+J%*8+7-O%:
      Y%=Y1%+Z%:GOSUB 3000
[VR2] 6140 DEF SEG=&HF000
[133] 6150 NEXT O%:NEXT Z%
[KR1] 6160 NEXT J%
[1W1] 6170 RETURN
```

Listing 1. Das Basic-Programm demonstriert die Initialisierung und Anwendung des 16-Farben-Modus der CGA-Farbgrafikkarte

```
nnnn:0100 1E      push DS      ;Basic-Datensegment sichern
nnnn:0101 B800B8  mov AX,B800  ;Segmentadresse des Videospeichers in AX
nnnn:0104 8ED8    mov DS,AX    ; und AX in DS kopieren
nnnn:0106 BF0000  mov DI,0000  ;Zielindex auf Anfang des Videospeichers
nnnn:0109 B9401F  mov CX,1F40  ;8000 Durchläufe
nnnn:010C B0DE    mov AL,DEH   ;ASCII-Zeichen 222 in AL
nnnn:010E 8805    mov [DI],AL  ; und in DS:[DI]
nnnn:0110 47      inc DI       ;Zielindex auf Offset des Attributs
nnnn:0111 B000    mov AL,00    ;Vorder- und Hintergrund schwarz
nnnn:0113 8805    mov [DI],AL  ;Attribut in die folgende Adresse
nnnn:0115 47      inc DI       ;Zielindex um Eins erhöhen
nnnn:0116 E2F4    loop 010C    ;insgesamt CX-mal wiederholen
nnnn:0118 1F      pop DS      ;Basic-Datensegment vom Stapel holen
nnnn:0119 CB      retf        ;zurück ins Basic-Programm
```

Listing 2. Diese Assemblerroutine steckt in den Daten des Basic-Programms

Programmautoren gesucht

Haben Sie Programme geschrieben, die noch nicht veröffentlicht wurden? Dann schicken Sie sie an die DOS-Redaktion. Wir suchen Utilities, Anwendungen und Tips und Tricks in Basic, Pascal, C oder einer anderen Programmiersprache, die unter MS-DOS auf dem PC eingesetzt wird.

Schicken Sie Ihr Programm auf einem Datenträger (5 1/4- oder 3 1/2-Zoll-Format) an

Redaktionsbüro Everts & Hagedorn
Redaktion »DOS International«
Hans-Stießberger-Str. 3
8013 Haar/München

Damit wir Ihr Programm mit Hilfe unseres Checksummers übersichtlich formatiert ausdrucken können, möchten wir Sie bitten – soweit möglich – die Breite der Programmzeilen im Quellcode auf maximal 44 Zeichen (mit Einrückungen) zu begrenzen.

In dem Begleitschreiben muß unbedingt der Interpreter beziehungsweise Compiler genannt werden. Vermerken Sie bitte auch Ihre Anschrift sowohl auf dem Anschreiben als auch auf der Diskette, so daß wir Ihnen im Falle einer Veröffentlichung Ihren Honorarscheck problemlos zuschicken können.

Die MS-DOS-Stapeldateien

Den meisten ist bekannt, daß MS-DOS-Befehle in »Stapeldateien« gespeichert und von dort abgearbeitet werden können. Hier erfahren Sie, welche vielfältigen Möglichkeiten Ihnen damit zur Verfügung stehen und daß sich dahinter sogar eine einfache »Programmiersprache« verbirgt, mit der Sie viele Ihrer Tätigkeiten am PC automatisieren können.

Zuerst verwundert vielleicht der Name »Stapeldatei«. Der Name kommt daher, daß alle darin enthaltenen Zeilen wie ein Stapel von Befehlen interpretiert und ausgeführt werden. Die bekannteste Stapeldatei ist die Startprozedur »autoexec.bat« des Betriebssystems MS-DOS. Sie wird mit dem Betriebssystem zusammen geliefert und kann natürlich nach Wunsch verändert werden, ebenso wie Sie selbst beliebige Prozeduren selbst herstellen können. Das einzige, was Sie dazu benötigen, ist ein zeilenorientierter Editor, der ASCII-Dateteilen erstellt.

Beim Ablauf einer Stapeldatei werden die einzelnen Zeilen wie vom Benutzer direkt eingegebene Befehle ausgeführt und zuvor am Bildschirm angezeigt. Will man die Anzeige unterdrücken, muß man als ersten Befehl in der Stapeldatei, das »Bildschirmmecho« mit

```
echo off
```

ausschalten. Sollen dennoch bestimmte Texte während der Ausführung angezeigt werden, so können sie beim Echo-Befehl explizit angegeben werden. Betrachten Sie Listing 1 (die Nummern »nn:« am Anfang dienen nur der Orientierung in der Abbildung und sind nicht Bestandteil der Stapeldatei): das Echo wird ausgeschaltet, der Text der zweiten Zeile wird jedoch angezeigt, das angegebene Unterverzeichnis wird eröffnet und Wordstar mit dem Befehl »ws« (das heißt mit seinem Programmnamen »ws.com«) aufgerufen. Am Ende der Arbeit mit Wordstar wird die Stapelprozedur in Zeile 5 fortgesetzt und mit der Rückkehr in das Hauptverzeichnis beendet.

Zum Verständnis der Arbeit einer Batch-Datei ein kleines Experiment mit der Prozedur von Listing 2, die im Laufwerk B vorhanden sein muß. Wie immer wird die Prozedur durch Angabe ihres Namens (ohne den Zusatz »bat«) gestartet, also für die Stapelprozedur namens »test.bat« durch die Eingabe von

```
test
```

Den Ablauf des Experiments sehen Sie in Bild 1. Öffnen Sie nach der ersten Pause-Meldung

Wenn bereit, eine Taste betätigen

das Laufwerk und drücken Sie eine Taste. Sie erhalten sofort die gezeigte Fehlermeldung, weil MS-DOS die nächste Stapeldateizeile Nummer 3 lesen will, aber es nicht kann. Erst nach dem Schließen des Laufwerkes und der Eingabe »w« für »wiederholen« geht es korrekt weiter.

Stapeldateien werden Zeile für Zeile neu gelesen und ausgeführt

Nun verstehen Sie auch, warum MS-DOS Sie manchmal auffordert, die Diskette mit der Stapeldatei wieder einzulegen, wenn Sie mitten in einem solchen Ablauf eine andere Diskette, die zum Beispiel ein Textverarbeitungsprogramm enthält, eingelegt haben. Und Ihnen wird auch klar, wieso (speziell bei PCs mit zwei Laufwerken) bei der Verarbeitung längerer Stapelprozeduren diese hektischen Geräusche auftreten: MS-DOS liest eine Zeile, erledigt die darin angegebene Arbeit mit anderen Dateien, muß zurück zur Stapeldatei für die nächste Zeile, muß wieder auf der anderen Diskette den Befehl ausführen – ein ständiges Hin und Her! Besonders bei langen Autoexec-Dateteilen, die viel einzurichten haben, ist das störend und langsam.

Nun haben Sie also festgestellt, wie eine Batch-Datei bearbeitet wird. Eine weitere Eigenheit ist, daß Sie in der Prozedur andere Programme aufrufen können (wie zum Beispiel »wordstar« in Listing 1) oder sogar andere Stapelprozeduren. Tippen Sie die Prozedur aus Listing 3 ein und eine weitere namens »yyyy.bat«, die sich nur mit einem Echo-Befehl meldet, etwa durch

```
B:\>test
B:\>echo Bitte Laufwerk öffnen!
Bitte Laufwerk öffnen!

B:\>pause
Wenn bereit, eine Taste betätigen. . .

Nicht bereit. Fehler
bei Lesen von Laufwerk B:
Abbruch, Wiederholen, Ignorieren? w

B:\>echo Hier geht es weiter!
Hier geht es weiter!

B:\>
```

Bild 1. Ablaufprotokoll der Testprozedur

```
echo Hier ist YYYY.bat
```

Sie werden feststellen, daß nach Beendigung keine Rückkehr in die rufende Prozedur erfolgt, deren Zeile 4 also nicht ausgeführt wird. Ein Hinweis für diejenigen, die eine Stapelprozedur wie ein Unterprogramm aufrufen wollen, also von der aufgerufenen Prozedur in die rufende zurückkehren wollen. In MS-DOS 3.3 gibt es dafür den Call-Befehl, aber wenn Sie eine ältere Version benutzen, brauchen Sie auch nicht zu verzagen. Es gibt auch hier verschiedene Tricks, aber die einfachste Lösung ist, eine Kopie des Kommando-Prozessors zu starten, wie Sie in Listing 4 (Zeile 3) sehen. Testen Sie es mit ihrer Unterprozedur »yyyy.bat«! Dazu muß allerdings der Kommando-Prozessor »command.com« des MS-DOS im angemeldeten Laufwerk verfügbar sein. Wer das nicht möchte und zusätzlich die Ausführung beschleunigen will, kopiert ihn mit auf die RAM-Disk mit dem Befehl

```
copy command.com c:
```

(falls »c« der Kennbuchstabe für die RAM-Disk) und fügt das RAM-Laufwerk in der Zeile mit dem Aufruf von »yyyy« ein:

```
c:command /c yyyy
```

Stapelprozeduren werden mächtig und wirkungsvoll durch die Übergabe von Namen oder anderen Zeichenketten als »Parameter«. Die Parameter werden in der Prozedur mit eins beginnend durchnummeriert und durch ein vorangestelltes »%« gekennzeichnet. Sie können in der Stapelprozedur beliebig oft angegeben werden. Betrachten Sie Listing 5 und tippen Sie es gleich ein. Wenn Sie damit experimentieren, werden die beiden Parameter »%1« und »%2« durch die beiden Zeichenketten ersetzt, mit der Sie die Prozedur aufgerufen haben – oder anders formuliert: Sie müssen die Prozedur mit ihrem Namen gefolgt von zwei weiteren (durch ein Leerzeichen getrennten) Angaben aufrufen, die natürlich auch Sinn machen müssen. In dem Beispiel ist »%1« offensichtlich ein zu dokumentierender Dateiname, »%2« das Ziel, nämlich entweder wieder eine Datei oder die Einheit »prn« für Drucker-

```
A:\>dok dok.bat con
A:\>echo ON
A:\>REM DOK.bat dokumentiert Textdateien
A:\>echo +++dok.bat+++ >con
+++dok.bat+++

A:\>type dok.bat >>con
echo ON
REM DOK.bat dokumentiert Textdateien
echo %1 >%2
type %1 >>%2
echo Datei %1 in "%2%" dokumentiert

A:\>echo Datei dok.bat in "con" dokumentiert
Datei dok.bat in "con" dokumentiert
```

Bild 2. Ablaufprotokoll der Prozedur mit Parametern


```
A:\>kopie xxx.bat xxx
A:\>echo off
Datei "xxx.bat" auf "xxx" kopiert.
A:\>kopie xxx.bat xxx
A:\>echo off
*** ACHTUNG *** Datei "xxx" schon vorhanden,
Abbruch mit [Ctrl-C], sonst Überschreiben.
Wenn bereit, eine Taste betätigen. . . ^C
Stapeljob beenden (J/N)? j
```

Bild 3. Ablaufprotokoll der Kopier-Prozedur

oder »con« für Bildschirm-Ausgabe. Testen Sie es mit

```
dok dok.bat con
```

(Bild 2), also die Dokumentation eben dieser Datei auf dem Bildschirm. Zur Verdeutlichung des Ablaufes ist das in einer Stapelprozedur übliche erste »echo off« abgeschaltet. Sie sehen in Bild 2, wie Ihre Parameter genau in dem Kontext ersetzt werden, in dem sie stehen. Der Echo-Befehl in Zeile 3 stellt den durch »+« herausgehobenen Namen der zu dokumentierenden Datei in die gewünschte Ausgabe »%2«, der folgende Type-Befehl fügt die Datei selbst hinten an.

Peinlich nur, wenn Sie beim Aufruf der Stapelprozedur einen Parameter vergessen! Er ist nicht vorhanden, also wird sein »Platzhalter« in der Form »%n« durch »nichts« im wahren Sinn des Wortes ersetzt. Der Aufruf

```
dok Notiz.txt
```

ohne zweiten Parameter macht also aus Zeile 3 einfach

```
echo +++Notiz.txt+++ >
```

und erzeugt damit eine Fehlermeldung, weil nach dem Datenumleitungszeichen »>« etwas stehen muß. Einige Besonderheiten der Parameterbehandlung sind noch von Bedeutung. Erstens werden sie in der Stapelprozedur überall (sogar auch in nicht auszuführenden Rem-Anweisungen) aufgrund ihres Aussehens »%n« erkannt und ersetzt, sie benötigen also keine weiteren Trennzeichen (auch nicht »Leerzeichen«). Zwei Zeilen einer Prozedur »x.bat« in der Form

```
copy %1 disk%2%3
echo Datei "%1" auf %3-Diskette Nr. %2
```

Befehl(skombination)	Wirkung
rem	markiert nicht auszuführende Kommentarzeilen
echo on/off	schaltet Ausführungsprotokoll ein/aus
echo »Kommentar«	zeigt den Kommentar am Bildschirm an
if »Bedingung« »Aktion«	führt die Aktion aus, wenn die Bedingung erfüllt ist
if not »Bed.« »Aktion«	Aktion, wenn Bed. nicht erfüllt ist
goto »xxx«	Sprung zu einer »Marke« xxx
:	Kennzeichen der Marke (z.B. »:xxx«)
exist »Datei«	stellt fest, ob Datei existiert (Sonderform der Bedingung bei »if«)
pause	unterbricht Ausführung, bis Taste betätigt
shift	verschiebt die Zuordnung der Parameter
for »Var.« do »Aktion«	wiederholt Aktion mehrmals, wie in »Var.« (einer Liste) angegeben
%n = »Wert«	Abfrage auf Parameter »gleich« Wert hinter »if«

Tabelle. Befehle oder Angaben in Stapelprozeduren

ergäben beim Aufruf mit

```
x read.me 027 A
```

als ausführbare Befehle die Zeilen

```
copy read.me disk027A
echo Datei "read.me" in A-Datei Nr. 027
```

Zweitens wird vom MS-DOS streng auf die Schreibweise der Parameter geachtet, also auch Groß- und Kleinschreibung unterschieden. Als nützliche Illustration in diesem Zusammenhang dient Listing 6, eine Stapelprozedur, die einen nicht seltenen und in der Regel katastrophalen Fehler verhindern soll, nämlich das Formatieren der Festplatte. Dazu entfernen Sie den Format-»Befehl« erst einmal vollständig von Ihrer Platte, indem Sie das entsprechende Programm einfach umbenennen:

```
ren c:format.com gefahr.com
```

Nun bauen Sie sich eine Prozedur namens »format.bat« wie in Listing 6, in der Sie deutlich sehen, daß die Laufwerk-Abfrage in Zeile 2 und 3 Groß- und Kleinschreibung getrennt abfragen muß. Die Formatierung der Festplatte wird dann schlichtweg abgelehnt, Sie müssen schon den gefährlichen Befehl

```
gefahr c:
```

direkt geben, um das zu tun. Alle anderen Formatierbefehle laufen wie bekannt, zum Beispiel

```
format a: /s
```

Soweit einige Besonderheiten über die Art und Weise, in der Parameter behandelt werden.

Die Stapelprozedur – »Programmiersprache« hat einige interessante Befehle

Wie sieht nun die »Programmiersprache« des MS-DOS aus? Die Elemente dieser Sprache finden Sie in der Tabelle.

Interessant sind die Anwendungen des »if« in Verbindung mit »exist«, der Existenzprüfung für Dateien. Wer hat sich nicht darüber geärgert, wenn er aufgrund eines Versehens mit

```
copy datei1 datei2
```

```
A:\>typeout kopie.bat
A:\>echo off
Drucken von "kopie.bat", bitte etwas Geduld...
-----
Dskt/Platte in Laufwerk A ist DISK 025 A
-----
kopie.bat
Datum ist: Mo. 14.03.1988
Zeit ist: 21.00.18,50
-----
echo off
if not exist %2 goto ok
echo *** ACHTUNG *** Datei "%2" schon vorhanden,
echo Abbruch mit [Ctrl-C], sonst Überschreiben.
pause
:ok
copy %1 %2 >nul
echo Datei "%1" auf "%2" kopiert.
```

Bild 4. Mit »typeout.bat« dokumentierte Prozedur

eine vorhandene wichtige »datei2« überschrieben und damit verloren hat, nur weil er nicht vorher geprüft hatte, ob »datei2« nicht schon vorhanden ist. Lassen Sie doch das Betriebssystem für Sie die Arbeit machen!

Listing 7 zeigt Ihnen den Weg dazu und Bild 3 den Lohn der Mühe. Sie sehen, daß in Zeile 02 zur Marke »ok« verzweigt wird, wenn die Zielfeile noch nicht existiert. Beim zweiten Durchlauf derselben Prozedur ist das der Fall, also wird die Warnung ausgegeben und die Meldung des Pause-Befehls (von der MS-DOS-Version abhängig) wird ausgegeben. Durch [Ctrl-C] wird die Stapelprozedur nach Rückfrage unterbrochen. Die Meldung des Copy-Befehls wurde durch »>nul« unterdrückt und in der folgenden Zeile durch eine eigene Meldung ersetzt.

Es sollten noch Maßnahmen erwähnt werden, um Ihre Prozeduren »abssturz sicher« zu machen, zum Beispiel durch Prüfung auf fehlende Parameter. Grundsätzlich sollte jede Prozedur, die ersetzbare Parameter verwendet (also in der Form »Name Parameter1 Parameter2 ...« aufgerufen wird), das Vorhandensein aller Parameter überprüfen und sogar als zusätzliche Dokumentationshilfe mit

```
Name ?
```

ihre Aufrufregeln anzeigen. Die Abfrage eines Parameters »%1« auf einen Wert erfolgt durch

```
if %1==Wert
```

Dabei ist allerdings zu beachten, daß %1 durch den Parameter beim Aufruf ersetzt wird. Geben Sie also

```
Name Par1
```

ein, so wird aus der Abfragezeile

```
if Par1==Wert
```

So können Sie abhängig von beim Aufruf mitgegebenen Werten den Ablauf der Stapelprozedur steuern, etwa über »goto«.

So weit, so gut. Was passiert jedoch, wenn der Parameter vergessen wird, die Prozedur mit »nichts« aufgerufen wird? Wie gesagt, für »%1« wird dann auch »nichts« eingesetzt; die Abfrage lautet also

```
if ==Wert
```


Dies führt wegen dieses ungültigen Formates zu einem Syntaxfehler. Verknüpfen Sie jedoch den Parameter mit einem beliebigen Zeichen, zum Beispiel in der Form

```
if %1==*Wert
```

oder auch

```
if "%1"=="Wert"
```

so passiert bei fehlendem Parameter nichts Schlimmes – die Bedingung ist nur einfach nicht erfüllt. Die Auswahl der Zusatzzeichen ist eine Frage des persönlichen Geschmacks, sollte aber »standardisiert« sein, das heißt nicht in jeder Stapelprozedur wechseln, um beim Lesen nicht zu viel Verwirrung zu stiften.

Standardisierter Prozeduraufbau erleichtert die Benutzung und Dokumentation

Deswegen sollte ein Prozedurkopf standardmäßig so aussehen wie in Listing 8, wo die Dokumentationsroutine aus Listing 5 ein wenig modifiziert wurde. Sie soll jetzt nur noch mit einem Parameter aufgerufen werden und die Ausgabe auf Bildschirm und Drucker anzeigen. Außerdem sollen Datum und Zeit mit ausgegeben werden, um jeweils den neuesten Stand erkennen zu können. In der zweiten Zeile programmieren Sie den »Fragezeichen-Aufruf«, der die Dokumentation der Aufrufregeln anzeigt und dann (Zeile 6) zum Ende der Befehle geht.

In der achten Prozedurzeile wird der besprochene Mechanismus benutzt, um das Fehlen des Parameters festzu-

stellen und eine entsprechende Fehlermeldung auszugeben. Danach kommt der eigentlich wesentliche und »Arbeit verrichtende« Teil der Stapelprozedur (wie so oft beim Programmieren: die Behandlung der Sonderfälle und die Dokumentation nimmt mehr Raum ein als der »produktive« Code): in Zeile 12 sehen Sie eine »if exist«-Kombination mit einem Befehl als »Aktion«, dann folgt eine Meldung an den Benutzer, um ihn bei Laune zu halten, denn die folgenden Befehle dauern einige Zeit. Hier können Sie all Ihr Wissen um Ausgabeumleitung und Filter einsetzen.

Zuerst wird in Zeile 14 aus der Dir-Ausgabe die Zeile mit dem Diskettenkennsatz herausgefiltert (das ist abhängig von der MS-DOS-Version), um zu dokumentieren, auf welchem Datenträger sich Ihre Datei befindet. Dann folgt ein dekorativer Strich und der Name der zu dokumentierenden Datei. In Zeile 17 und 18 werden die gewünschten Angaben aus Datum und Zeit herausgefiltert. Schließlich wird nach einem weiteren Strich noch die Datei »%1« selbst angehängt und dann am Bildschirm und auf dem Drucker präsentiert (Bild 4).

Ein weiterer Befehl, der nur in Stapelprozeduren zur Verfügung steht, ist »shift«. Schauen Sie sich auch hier gleich wieder eine Nutzenanwendung an (Listing 9 und Bild 5). Die Zahl der Parameter ist auf 10 begrenzt, außerdem hat ja jeder Parameter seine feste Zuordnung: die erste im Aufruf angegebene Zeichenfolge wird zu »%1«, die zweite zu »%2«

und so weiter.

Wollen Sie eine beliebige Anzahl von Parametern übergeben oder die Zuordnung verschieben, so hilft der Befehl »shift«, denn er »vergißt« den ersten Parameter, macht aus »%2« den ersten »%1«, setzt den bisherigen dritten »%3« auf »%2« und behandelt so alle im Aufruf angegebenen Zeichenketten. In Verbindung mit der Überprüfung auf »leer« und entsprechenden Sprungbefehlen wird daraus in Listing 9 eine nützliche Stapelprozedur, die genausoviel Dateien in ein Platten-Unterverzeichnis »archiv« kopiert, wie im Aufruf angegeben sind. Dabei wird »xxx« zu »%1«, »xxx.bat« zu »%2« und »yyyy.bat« zu »%3«. In der Prozedur wird nur »%1« angesprochen, und bei jedem »shift« wird der folgende Parameter dorthin übertragen. Beim vierten Shift-Befehl ist »%1« leer, und die Prozedur wird beendet.

Eine richtige Programmschleife erweitert die Möglichkeiten einer Stapelprozedur

In diesem Beispiel war die Schleife »zu Fuß« programmiert, das heißt über eine If-Abfrage und Goto-Befehle. MS-DOS sieht aber auch eine »echte« For-Schleife vor, in der eine »Liste« (in der Bedeutung von »Aufzählung«) abgearbeitet werden kann. Die Syntax (in der Tabelle schon skizziert) lautet vollständig:

```
for %Variable in (Liste) do Befehl
```

Dabei spielt der Name der Variablen keine Rolle, er dient nur dazu, daß man auf ihren aktuellen Wert Bezug nehmen kann, denn dieser Wert ist nacheinander jedes Element der Liste. Ein praktisches Beispiel ist das Umwandeln eines Assembler-Quellprogramms »pgm«, denn dabei

```
A:\>archiv xxx xxx.bat yyyy.bat
A:\>echo off
Datei "xxx" archiviert.
Datei "xxx.bat" archiviert.
Datei "yyyy.bat" archiviert.
```

Bild 5. Protokoll des Archivierungsvorganges

```
A:\> listall d*.bat
A:\>echo off
--- Dateien in "d*.bat" sind:
DISKNEU.BAT
DOK.BAT
DOKALL.BAT
```

Bild 6. Ablaufprotokoll von »listall«

Das universelle Programmiersystem mc – AllProg II mit neuen Möglichkeiten

Universell,
da eigenständig !

Keine Einsteckkarte!
Passt an IBM PC XT AT
PS/2 und kompatible
sowie
Atari ST und weitere!

PALs und CMOS-PALs
von TI, AMD/MMI, NS, ...
GALs von Lattice, SGS, ...
EPLDs von Altera, Intel, ...
EPROMs, EEPROMs, Single-
Chip, AkkuRAMs
PLDs von Signetics/Valvo
PROMs von Valvo, TI, NS,...

Komfortable menügesteu-
erte Bedienerführung.
Verarbeitet Intel-Hex,
Motorola-S, Binär, JEDEC.
Download-Möglichkeit für
Programmier Routinen bei
Fertiggeräten (Sie können
eigene Software einbinden)

Fertiggeräte oder Kompo-
nenten ab 997,- DM,
Adaptersockel für SMT und
SingleChip, Zubehör sowie
ausführliche

Informationen
bei:

H. Haase & M. Menrad GbR Computerperipherie, Postfach 2647, 3300 Braunschweig, Tel. 0531/400310

müssen Sie ja folgende drei Schritte ausführen: umwandeln, binden und in eine Com-Datei wandeln. Diese Befehle müßten Sie eingeben:

```
masm pgm
link pgm
exe2bin pgm
```

Eine Stapelprozedur »assembl.bat« erledigt die drei Schritte mit nur einer Zeile:

```
for %i in (masm link exe2bin) do %i %1
Rufen Sie sie mit
assembl pgm
```

auf, so passiert folgendes: »%i« (ein Name, den man häufig dazu verwendet; er ist der Bezeichnung von Laufvariablen aus Fortran entlehnt) erhält nacheinander die Werte aus der Liste in Klammern, hat also zuerst den Inhalt »masm«, dann »link« und schließlich »exe2bin«.

Mit der For-Schleife lassen sich Arbeitsabläufe sehr wirkungsvoll vereinfachen

Jedesmal wird damit der auf das »do« folgende Befehl ausgeführt, der aus der Kombination des jeweiligen Inhaltes von »%i« mit dem Parameter aus dem Aufruf besteht und damit genau die zuvor angegebenen drei Befehle für die Programmdatei »pgm« erzeugt. Interessant und nützlich ist, daß die »Liste« nicht nur aus vorge-

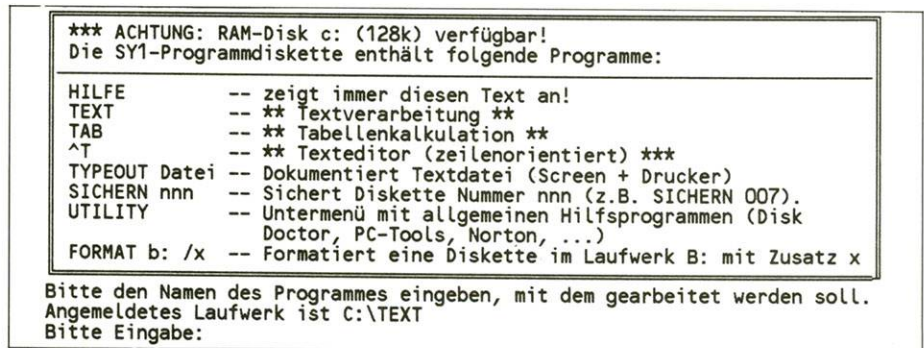


Bild 8. Hilfe- bzw. Auswahlanzeige der Menüsteuerung

gebenen Teilen bestehen darf, sondern daß das MS-DOS auch eine Wildcard-Angabe als Liste ansieht, also etwa *.txt« als Liste aller Dateinamen, die auf *.txt« enden. Eine solche Liste darf im For-Befehl angegeben werden. Experimentieren Sie mit Listing 10, einer Prozedur, die zum Beispiel mit

```
listall *.txt
```

aufgerufen werden kann. Die Variable »%i« nimmt nacheinander jeden Wert der Liste an, sodaß der nach dem »do« folgende Echo-Befehl jede einzelne Datei dieser Liste anzeigt (vergleichen Sie Bild 6).

Hinter dem »do« darf jede beliebige Anweisung stehen, nur kein weiteres »for« und nur genau eine Anweisung.

Sollen jedoch mehrere Anweisungen ausgeführt werden, müssen Sie das MS-DOS überlisten: Sie packen sie in eine Stapelprozedur und rufen sie hinter dem »do« auf. Dabei müssen Sie bedenken, daß ein Aufruf einer Prozedur von einer anderen heraus ohne Rückkehr stattfindet. Also muß – wie dort ebenfalls besprochen – ein zweiter Kommandoprozessor aufgerufen werden. Damit schreiben Sie also eine Stapelprozedur »dokall.bat«, die eine zu diesem Zweck leicht modifizierte Dokumentationsroutine »dok.bat« aufruft. Den kleinen aber feinen Unterschied zu Listing 5 finden Sie sicher selbst heraus.

In Bild 7 ist (zur Verdeutlichung des Abarbeitens der Liste ohne Echo-off-Befehl) der Ablauf von »dokall« abgedruckt, wobei praktischerweise gleich die beiden fraglichen Stapelprozeduren selbst dokumentiert wurden. Es ist zur Trennung von Protokoll und Ergebnis sinnvoll, hier als Ziel eine Datei wie etwa »dobat.txt« anzugeben, deren Inhalt Sie dann mit dem Type-Befehl bequem ansehen können.

Durch geschickte Anwendung von Stapeldateien lassen sich auch komplexe Menüsysteme erzeugen

Eine weitere Anwendungsmöglichkeit von Stapeldateien ist eine Art »Menüsteuerung«. Dabei wird eine Textdatei namens »menue«, wie sie in Bild 8 angedeutet ist, am Bildschirm angezeigt, zum Beispiel mit dem Befehl

```
type menue
```

als letztem in Ihrer »autoexec.bat«. Dieser Type-Befehl ist auch einziger Inhalt der »hilfe.bat«. Die Textdatei enthält übrigens die letzten beiden Zeilen aus dem Listing nicht, endet also mit

Bitte den Namen ... soll

Die letzten beiden Zeilen werden durch einen modifizierten Prompt-Befehl erzeugt, den Sie vorher in der Startprozedur untergebracht haben:

```
A:\>dokall do*.bat dobat.txt
A:\>REM DOKall.bat dokumentiert Textdateien aus do*.bat-Liste
A:\>del dobat.txt
A:\>for %i in (do*.bat) do c:command /c dok %i dobat.txt
A:\>c:command /c dok DOK.BAT dobat.txt
A:\>echo off
Datei DOK.BAT in "dobat.txt" dokumentiert
A:\>c:command /c dok DOKALL.BAT dobat.txt
A:\>echo off
Datei DOKALL.BAT in "dobat.txt" dokumentiert
A:\>
A:\>type dobat.txt
-----
+++DOK.BAT+++
echo off
REM DOK.bat dokumentiert Textdateien
echo ----- >%2
echo +++%1+++ >>%2
type %1 >>%2
echo Datei %1 in "%2%" dokumentiert
-----
+++DOKALL.BAT+++
REM DOKall.bat dokumentiert Textdateien aus %1-Liste
del %2
for %i in (%1) do c:command /c dok %i %2
A:\>
```

Bild 7. Modifizierte Dokumentationsroutinen und ihr Ablauf

prompt Angemeldetes Laufwerk ist \$p
\$_Bitte Eingabe:

Er erzeugt die gezeigte zweizeilige Systemmeldung. Durch die Eingabe von ANSI-Steuersequenzen in Ihrer Datei »menue« können Sie Bildschirmfarben und Cursorposition beliebig steuern.

Es bleiben die im Menü angegebenen »Befehle« zu realisieren – auch sie sind selbstverständlich Stapelprozeduren, einschließlich des im Hauptmenü erwähnten Untermenüs für die Utilities. So wäre zum Beispiel Ihre »text.bat« genau das Beispiel aus Listing 1, erweitert um eine sechste Zeile

type menue

um das Hauptmenü nach Beendigung der Textverarbeitung wieder am Bildschirm anzuzeigen. Eine solche komfortable Menüsteuerung läßt sich individuell auf die jeweilige Situation zuschneiden. Sie läuft am besten auf einem schnellen Speichermedium, etwa einer Festplatte oder besser der RAM-Disk. Aber auch mit nur zwei Diskettenlaufwerken haben Tüftler hier schon Erstaunliches zuwege gebracht. Alles was Sie brauchen, sind Einfallsreichtum und solide Grundkenntnisse der Wirkungsmechanismen von Stapelprozeduren. Nun haben Sie sicher reichlich Ideen bekom-

men, wie Sie mit Ihrer Kenntnis der Funktionsweise von Stapelprozeduren Ihre Arbeit effektiver gestalten können. In der Praxis werden Sie bei allen Ihren Stapelprozeduren den Standard-Kopf nach dem Beispiel von Listing 7 einsetzen, um den Aufruf zu dokumentieren und fehlende Parameter mit einer passenden Fehlermeldung abzufangen. Je intensiver Sie mit Prozeduren arbeiten, desto interessantere Anwendungen werden Sie finden und sich nach und nach einen »Werkzeugkasten« von nützlichen Stapeldateien aufbauen.

(Jürgen Beetz/dm)

```
[DA1] 1: echo off
[RU9] 2: echo Aufruf WORSTAR im Unterverzeichnis \TE
      XT\WORDSTAR
[U23] 3: cd \text\wordstar
[7D0] 4: ws
[EH0] 5: cd \
```

Listing 1. Einfache Stapelprozedur zum Aufruf von Wordstar

```
[1N4] 1: echo Bitte Laufwerk öffnen!
[MW0] 2: pause
[PY3] 3: echo Hier geht es weiter!
```

Listing 2. Test-Prozedur »test.bat«

```
[KE5] 1: echo Hier ist die Hauptprozedur!
[KN4] 2: echo Aufruf von "yyyy.bat":
[UV0] 3: yyyy
[YF7] 4: echo Hier geht es in der Hauptprozedur weiter!
```

Listing 3. Aufruf einer Prozedur aus einer anderen heraus

```
[8P4] 1: Hier ist die Hauptprozedur!
[KN4] 2: echo Aufruf von "yyyy.bat":
[HN2] 3: command /c yyyy
[YF7] 4: echo Hier geht es in der Hauptprozedur weiter!
```

Listing 4. Aufruf einer Prozedur als »Unterprogramm«

```
[DA1] 1: echo off
[GB6] 2: REM DOK.bat dokumentiert Textdateien
[522] 3: echo ++%1+++ >%2
[7P1] 4: type %1 >>%2
[1A5] 5: echo Datei %1 in "%2%" dokumentiert
```

Listing 5. Eine Prozedur mit »Parametern«

```
[DA1] 1: echo off
[Y33] 2: if %1==*C: goto nein
[Y33] 3: if %1==*c: goto nein
[9L1] 4: gefahr %1 %2
[5D1] 5: goto end
[RU0] 6: :nein
[BB0] 7: echo *** Festplatte kann nicht formatiert werden ***
[ZN0] 8: :end
```

Listing 6. Neuer Formatier – »Befehl«: »format.bat«

```
[DA1] 1: echo off
[6P3] 2: if not exist %2 goto ok
[W27] 3: echo *** ACHTUNG *** Datei "%2" schon vorhanden,
```

```
[Z28] 4: echo Abbruch mit [Ctrl-C], sonst Überschreiben.
[MW0] 5: pause
[DM0] 6: :ok
[882] 7: copy %1 %2 >nul
[ES4] 8: echo Datei "%1" auf "%2" kopiert.
```

Listing 7. Kopier-Prozedur »kopie.bat«

```
[DA1] 1: echo off
[VH3] 2: if not %1==*? goto exec
[B4A] 3: echo Textdatei wie mit TYPE dokumentieren, OUT auf Drucker
[WM6] 4: echo mit Datum/Zeit + Disketten-Kennung
[ED9] 5: echo Aufruf: TYPEOUT Datei.Ext (1 Datei, z.B. AUTOEXEC.BAT)
[5D1] 6: goto end
[EV0] 7: :exec
[Y43] 8: if not %1==* goto run
[PH8] 9: echo *** FEHLER ! *** Aufruf TYPEOUT Datei.Ext (1 Datei!)
[5D1] 10: goto end
[CS0] 11: :run
[YV3] 12: if exist c:xxx del c:xxx
[YR6] 13: echo Drucken von "%1", bitte etwas Geduld..
[XG4] 14: dir | c:find "Dskt/" >c:xxx
[SG6] 15: echo -----
      --- >>c:xxx
[9F2] 16: echo %1 >>c:xxx
[A55] 17: date <cr | c:find "ist" >>c:xxx
[665] 18: time <cr | c:find "ist" >>c:xxx
[SG6] 19: echo -----
      --- >>c:xxx
[ZK2] 20: type %1 >>c:xxx
[TT1] 21: type c:xxx
[5P2] 22: type c:xxx >prn
[ZN0] 23: :end
```

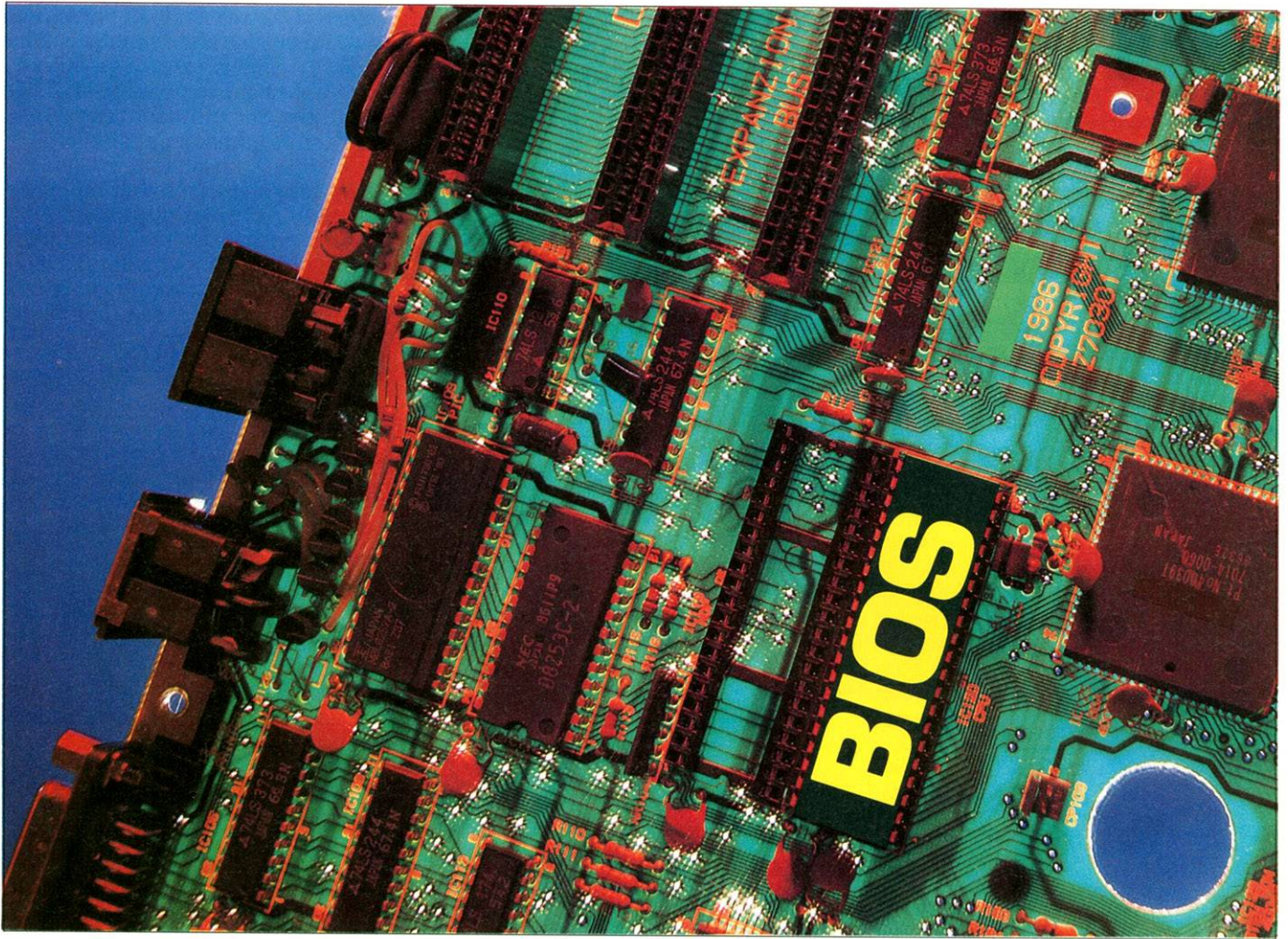
Listing 8. Standardisierte Stapelprozedur zur Dokumentation

```
[DA1] 1: echo off
[3W0] 2: :loop
[XF2] 3: if %1==* goto end
[UH3] 4: copy %1 c:\archiv >nul
[L64] 5: echo Datei "%1" archiviert.
[GW0] 6: shift
[9L1] 7: goto loop
[ZN0] 8: :end
```

Listing 9. Stapelprozedur zum Archivieren beliebig vieler Dateien

```
[DA1] 1: echo off
[9S7] 2: REM Listall.bat zeigt Textdateien aus %1-Liste
[2C4] 3: echo --- Dateien in "%1" sind:
[MV3] 4: for %i in (%1) do echo %i
```

Listing 10. Test-Prozedur »listall.bat«



Das BIOS: einfach unersetzlich

Sobald Sie Ihren Computer einschalten, beginnt der Prozessor mit der Ausführung der im BIOS enthaltenen Routinen. Doch neben dem Hardwareselbsttest enthält das BIOS auch die Funktionen für den Zugriff auf die verschiedenen Peripherieeinheiten wie Bildschirm, Tastatur, Schnittstellen und Datenträger.

Die Abkürzung BIOS steht für »Basic Input Output System«, was auf Deutsch soviel wie »grundlegendes Ein-/Ausgabe-System« bedeutet. In jedem PC, AT oder 386er PC ist das BIOS in einem EPROM (ein Festwertspeicher-Baustein) abgelegt, der sich am Ende des durch MS-DOS adressierbaren 1 MByte großen Speicherbereichs befindet. Die Größe des EPROM ist je nach Computertyp und Hersteller verschieden, beträgt in der Regel jedoch 16 oder 32 KByte.

Nach dem Einschalten oder nach einem Reset beginnt der Prozessor mit der Ausführung der Maschinenbefehle ab der BIOS-Adresse FFFF:0000 hex. Dort steht gewöhnlich ein Sprungbefehl zur eigentlichen Initialisierungsroutine. Diese Routine programmiert als eine der ersten Hand-

lungen den Timer-Baustein und den DMA-Controller des PC so, daß sie die dynamischen Speicherbausteine in regelmäßigen Abständen wieder auffrischen.

Danach testet und initialisiert die Routine die übrigen Bausteine auf der Mutterplatine und bildet eine Prüfsumme für das BIOS. Wenn das BIOS dabei Fehler feststellt, sei es, daß sich Bausteine nicht korrekt initialisieren lassen, oder daß die Prüfsumme des BIOS inkorrekt ist, informiert es den Anwender darüber mit einer Fehlermeldung.

Darauf ermittelt das BIOS die Systemkonfiguration des Computers, wobei es dazu bei einem PC die Stellung der Dipschalter über den Parallel-Portbaustein 8255 abfragt und bei einem AT oder 386er PC auf den

nichtflüchtigen CMOS-Speicherbereich der Echtzeituhr zugreift. Von diesen Werten ausgehend, sorgt das BIOS für die korrekte Initialisierung des Bildschirmadapters und der anderen Erweiterungskarten. Danach sucht es nach seriellen und parallelen Schnittstellen und legt die so gefundenen Portadressen im Datensegment des BIOS ab der Speicheradresse 00400hex ab.

Um eine größere Flexibilität bei der Entwicklung späterer Computermodele zu erreichen, verwendeten die IBM-Entwickler ein für zukünftige Änderungen und Erweiterungen geeignetes Konzept zur Ansteuerung der Peripherieeinheiten.

Statt die Routinen für den Zugriff auf die Peripherie an festgelegten Adressen abzulegen, die dann vom Betriebssystem oder von einem Anwendungsprogramm aufgerufen werden, zeigen die Interrupt-Vektoren 10hex bis 1Ahex auf diese Routinen. Dadurch ist es also im Prinzip völlig unerheblich, an welcher absoluten Adresse im BIOS die Routinen zur

Ansteuerung der Peripherie stehen, solange ein bestimmter Interrupt-Vektor auf sie zeigt.

Für den Aufruf einer BIOS-Routine löst das jeweilige Programm über den Assemblerbefehl »int nn« einen Software-Interrupt aus, wobei »nn« für die Nummer des Interrupts steht. Dabei lassen sich über die Prozessorregister sowohl Funktionen (mit dem AH-Register) und Unterfunktionen (mit dem AL-Register) auswählen, als auch verschiedene Parameter und Zeiger auf Parametertabellen übergeben.

Im PC werden die Peripherieeinheiten über die Vektoren der Software-Interrupts angesteuert

In der Initialisierungsroutine setzt das BIOS die verschiedenen Interrupt-Vektoren auf die entsprechenden Funktionen. Tabelle 1 zeigt alle Interrupts des PC, darunter auch die, deren zugehörige Funktionen im BIOS liegen. Dabei sind die Interrupts 00hex bis 07hex von Intel für den Prozessor reserviert und dienen nicht etwa zum Aufruf von Funktionen, sondern für die Reaktion auf schwere Hardware-Fehler und ähnliches (auch wenn sich IBM und Amstrad nicht an diese Konvention halten).

Der Interrupt 05hex wird durch die Tastenkombination [Shift-PrtSc] ausgelöst und gibt den aktuellen Inhalt des Bildschirms auf dem Drucker aus. Im BIOS selbst ist nur die Routine zum Ausdruck des Textbildschirms enthalten. Um einen Grafikbildschirm auszudrucken, müssen Sie zuvor das Programm »graphics.com« von Ihrer MS-DOS-Systemdiskette laden, das diese BIOS-Routine durch eine eigene ersetzt. Die Interrupts 08hex bis 0Fhex werden in der Regel von den Erweiterungskarten über den Interrupt-Controller auf der Mutterplatine ausgelöst, wenn eine entsprechende Steckkarte die Aufmerksamkeit des Prozessors benötigt.

Die Interrupts 10hex bis 1Ahex sind für den Zugriff auf die Peripherieeinheiten des PC wie Bildschirm, Tastatur, Schnittstellen und Datenträger zuständig, die in modernen PCs oft schon auf der Mutterplatine integriert sind.

Das BIOS setzt die Interrupt-Vektoren 1Bhex und 1Chex auf Dummy-Routinen, das heißt, diese Routinen bewirken nichts außer einem Rücksprung zum aufrufenden Programm. Hier kann ein Betriebssystem wie MS-DOS oder der Anwender später seine eigenen Funktionsaufrufe einbinden. Die drei verbleibenden BIOS-Inter-

rupt-Vektoren 1Dhex bis 1Fhex weisen nicht auf ausführbaren Programmcode, sondern auf Parametertabellen für Video- und Disketten-Controller sowie einen nachladbaren Grafikzeichensatz.

Nach der Initialisierung der Interrupt-Vektoren testet das BIOS den gesamten Arbeitsspeicher – je nach BIOS-Version mehr oder weniger gründlich. Dieser RAM-Test wird aber nur dann durchgeführt, wenn es sich um einen sogenannten Kaltstart handelt, also nach dem Einschalten oder nach einem Hardware-Reset. Ein Warmstart wird durch die Tastenkombination [Ctrl-Alt-Del] ausgelöst. Damit das BIOS unterscheiden kann, ob ein Speichertest notwendig ist, liest es die Speicherzellen 0040:0072 und 0040:0073hex aus. Nach erfolgreichem Speichertest schreibt das BIOS dort den Wert 1234hex hinein, des weiteren wiederholt der Tastaturtreiber dies, wenn er die Tastenkombination [Ctrl-Alt-Del] erkennt. Danach ruft der Tastaturtreiber die Initialisierungsroutine ab Adresse FFFF:0000hex auf.

Das BIOS testet über zwei Speicherzellen, ob ein Kalt- oder Warmstart erfolgen soll

Findet das BIOS bei einem Reset über [Ctrl-Alt-Del] nun das Datenwort 1234hex in den genannten Speicherzellen vor, geht es davon aus, daß bereits ein Speichertest stattgefunden hat, und verzichtet auf eine erneute Prüfung. Die meisten neueren BIOS-Versionen erlauben zudem den Abbruch des Speichertests durch eine beliebige Taste.

Nach diesem Speichertest sucht das BIOS nach eventuell weiteren vorhandenen Festwertspeichern, wie zum Beispiel die in einem EPROM abgelegte Treibersoftware für den Festplatten-Controller oder das erweiterte BIOS der EGA- und VGA-Karte. Dabei geht das BIOS vor folgend vor: Zunächst wird der Speicherbereich oberhalb der Adresse C0000hex nach den beiden aufeinanderfolgenden Kennung-Bytes 55 und AAhex durchsucht. Die Byte-Kombination 55AAhex kennzeichnet nämlich ein Erweiterungs-EPROM.

Findet das BIOS diese Kennung, dann geht es davon aus, daß das nächste Byte die Länge des zusätzlichen Programmcodes in 512 Byte-Schritten enthält. Daraufhin bildet es die Prüfsumme für diesen Speicherbereich. Dabei wird der Inhalt aller Bytes ohne Übertrag bei Verwendung eines 8-Bit-Registers addiert, so daß also beispielsweise die Werte 01hex, FEhex und 01hex den Wert Eins ergeben.

Die Summe dieser übertraglosen Addition muß Null sein, andernfalls meldet das BIOS einen Prüfsummenfehler. Aus diesem Grund sehen die Programmierer bei der Entwicklung einer BIOS-Erweiterung eine Speicherzelle vor, die den notwendigen Korrekturwert aufnimmt, damit die Summe nach der Addition aller Speicherzellen auch wirklich Null beträgt.

Wenn die Prüfsummenbildung einer BIOS-Erweiterung korrekt abgeschlossen ist, setzt der Prozessor die Programmausführung ab Offset 03 in dieser BIOS-Erweiterung fort, also direkt nach dem Byte, das die Länge des Erweiterungsspeichers enthält. So hat die BIOS-Erweiterung die Gelegenheit, Interrupt-Vektoren auf eigene Routinen umzuleiten, um zum Beispiel zusätzliche Funktionen einzubauen.

Jeder zusätzliche EPROM-Baustein muß seinem Programmcode den beschriebenen Identifikationskopf voranstellen, um automatisch initialisiert zu werden. Das BIOS setzt die Suche nach Erweiterungs-EPROMs maximal bis zur Adresse F600:0000hex fort, denn dort vermutet es einen ROM-residenten Basic-Interpreter. Allerdings gilt dies nur für das BIOS im IBM PC und hochgradig kompatible BIOS-Versionen. Abgesehen von den IBM Computern befindet sich nämlich normalerweise kein Basic-Interpreter in Erweiterungs-EPROMs.

Für das Zwischenspeichern von Daten verwendet das BIOS den Speicherbereich von 0040:0000 bis 0040:00FFhex. Die genaue Verwendung der einzelnen Adressen entnehmen Sie der Tabelle 2.

Bis zu diesem Punkt gibt es kaum Unterschiede zwischen dem BIOS in einem PC und dem in einem AT – sieht man einmal davon ab, daß die Diagnose-Routinen wegen der unterschiedlichen Hardware natürlich oft nicht identisch sind.

Der wesentlichste Unterschied zwischen dem BIOS des PC und AT ist der erweiterte Kassetten-Interrupt 15hex. Während er bei einem PC heutzutage ohne Funktion ist, weil niemand mehr einen Kassettenrecorder an seinem PC betreibt, hat er beim AT wieder Bedeutung erlangt. Um die zusätzlichen Funktionen des AT unterzubringen, haben die AT-Entwickler nämlich den wertlosen Kassetten-Interrupt um diese Funktionen erweitert. So läuft zum Beispiel der Zugriff auf den Arbeitsspeicher oberhalb der 640-KByte-Grenze oder die Umschaltung in den Protected Mode über den Interrupt 15hex. Vor dem Aufruf der Funktionen des Kassetten-Interrupt 15hex ist es allerdings sinnvoll, zu überprüfen, ob es

Die Software-Interrupts des PC	
Interruptnummer (hex)	Bedeutung
Prozessor-Interrupts	
00	Division durch Null
01	Einzelschrittmodus
02	Nicht maskierbarer Interrupt (NMI), der bei Paritätsfehlern aufgerufen wird
03	Abbruchpunkt (Breakpoint)
04	Überlauf bei Into-Befehl
05	Bildschirmhardcopy an Drucker senden (wird durch [Shift-PrtSc] aufgerufen) Speicherzelle 00500 = 0: korrekt beendet Speicherzelle 00500 = 1: Hardcopy wird gedruckt Speicherzelle 00500 = FFhex: Fehler aufgetreten
06	Unbenutzt (Amstrad PC 1512 und 1640: Aufruf einer Mausfunktion)
07	Unbenutzt
Interrupt-Controller	
08 (IRQ0)	Timer-Interrupt, wird 18,2 mal pro Sekunde durch den Zeitgeber-Baustein 8253 ausgelöst
09 (IRQ1)	Tastatur-Interrupt, wird bei jedem Tastendruck ausgelöst
0A (IRQ2)	Interrupt durch EGA-Karte
0B (IRQ3)	Interrupt durch die zweite serielle Schnittstelle (»com2«)
0C (IRQ4)	Interrupt durch die erste serielle Schnittstelle (»com1«)
0D (IRQ5)	Interrupt durch den Festplatten-Controller
0E (IRQ6)	Interrupt durch den Disketten-Controller
0F (IRQ7)	Interrupt durch den Drucker (»prn«, »lpt1«, »lpt2« oder »lpt3«)
BIOS-Interrupts	
10	Aufruf einer Video-Funktion (Funktionsnummer wird im AH-Register übergeben)
11	Ermittle Systemkonfiguration des Computers und übergebe Ergebnis im AX-Register Bit 0 = 1: Diskettenlaufwerk vorhanden Bit 1 = 1: Arithmetikprozessor 8087 vorhanden Bit 2 und 3: Speicher auf der Hauptplatine in 16-KByte-Einheiten Bit 5 = 0 und Bit 4 = 0: Bit 5 = 0 und Bit 4 = 1: 40x25 Zeichen (Farbgrafikkarte) Bit 5 = 1 und Bit 4 = 0: 80x25 Zeichen (Farbgrafikkarte) Bit 5 = 1 und Bit 4 = 1: 80x25 Zeichen (Monochromadapter) Bit 6 und 7: Anzahl der Diskettenlaufwerke (00 = 1, 01 = 2 und so weiter) Bit 8: – Bit 9 bis B: Anzahl der seriellen RS 232C-Schnittstellen Bit C = 1: Spieladapter vorhanden Bit D: – Bit E und F: Anzahl der angeschlossenen Drucker
12	Ermittle Größe des installierten Arbeitsspeichers (ohne Extended Memory) und übergebe Ergebnis (in KByte) im AX-Register
13	Aufruf einer Disketten- oder Festplatten-Funktion (Funktionsnummer wird im AH-Register übergeben)
14	Aufruf einer Funktion für die serielle Schnittstelle (Funktionsnummer wird im AH-Register übergeben)
15	Aufruf einer Kassettenrecorder-Funktion (PC) oder Zusatz-Funktion (AT) (Funktionsnummer wird im AH-Register übergeben) Schneider PC: Aufruf einer erweiterten BIOS-Funktion (batteriegepufferter RAM (NVR) und Grafik-Bitplanes für Grafikmodus 2)
16	Aufruf einer Tastatur-Funktion (Funktionsnummer wird im AH-Register übergeben)
17	Aufruf einer Drucker-Funktion (Funktionsnummer wird im AH-Register übergeben)
18	Start des ROM-residenten Kassetten-Basic ab Adresse F600hex (sofern vorhanden)
19	Lade Bootsektor von Sektor 0 auf Spur 0 ab Adresse 07C00hex und rufe ihn auf
1A	Aufruf einer Systemuhr-Funktion (Funktionsnummer wird im AH-Register übergeben)
Anwender-Interrupts	
1B	Abbruchanforderung über die Tastatur ([Ctrl-C] oder [Ctrl-Break])
1C	Ticker-Interrupt (wird 18,2 mal pro Sekunde durch den Interrupt 08 aufgerufen)
BIOS-Parameter	
1D	Zeiger auf den Anfang der Parameter-Tabelle für den Video-Controller
1E	Zeiger auf den Anfang der Parameter-Tabelle für den Disketten-Controller
1F	Zeiger auf den Anfang des Grafikzeichensatzes für ASCII-Zeichen 128 bis 255
MS-DOS-Interrupts	
20	Beendigung des laufenden Programms
21	Aufruf einer MS-DOS-Funktion (Funktionsnummer wird im AH-Register übergeben)
22	Adresse, die bei Beendigung des laufenden Programms aufgerufen wird
23	Adresse, zu der beim Auftreten von [Ctrl-C] verzweigt wird
24	Adresse, zu der bei Behandlung kritischer Fehler verzweigt wird
25	Diskettensektoren wahlfrei lesen AL = 0: Laufwerk A, AL = 1: Laufwerk B und so weiter CX = Zahl der Sektoren DX = erster Sektor (DS:BX) = Lesepuffer (DTA)

Tabelle 1. Ein Großteil der Software-Interrupts des PC weist auf Adressen im BIOS (Basic Input Output System)

MS-DOS-Interrupts	
26	Diskettensektoren wahlfrei schreiben AL = 0: Laufwerk A, AL = 1: Laufwerk B und so weiter CX = Zahl der Sektoren DX = erster Sektor
27	(DS:BX) = Schreibpuffer (DTA) Programm beenden und für Neustart im Arbeitsspeicher behalten CS:DX = Zeiger auf Programmende + 1
28 bis 3F	Für MS-DOS reserviert
40	Für Erweiterungen reserviert
41	Zeiger auf den Anfang der Parameter-Tabelle für den Festplatten-Controller
42 bis 5F	Für Erweiterungen reserviert
60 bis 67	Für Anwender-Interrupts reserviert
68 bis 7F	Nicht verwendet
80 bis 85	Für Basic reserviert
86 bis F0	Basic-Interpreter-Interrupts
F1 bis FF	Nicht verwendet

Fortsetzung Tabelle 1

sich bei dem vorhandenen Rechner überhaupt um einen AT handelt. Dazu gibt es mehrere Verfahren. Das einfachste ist die Auswertung des Rechnertyp-Byte, das unter der Adresse F000:FFFEhex ausgelesen wird. Dieses Byte gibt den Typ des verwendeten Rechners an. Eindeutig definiert ist dieses Byte allerdings nur bei den IBM Computern selbst. Die Werte des Rechnertyp-Byte haben folgende Bedeutung:

252: IBM AT
254: IBM XT
255: IBM PC

Leider haben sich die Hersteller der Kompatiblen nicht ganz an dieses Vorbild gehalten. Lediglich bei ATs und 386er PCs darf man davon ausgehen, daß das Rechnertyp-Byte den Wert 252 enthält, wohingegen das BIOS so mancher PCs eine herstellereigene Modellidentifikation angibt.

Mit ein paar einfachen Peek-Befehlen lassen sich dem BIOS Informationen entlocken

In GW-Basic läßt sich der Rechnertyp durch Auswertung des Rechnertyp-Byte auf folgende Weise bestimmen:

```
10 DEF SEG=&HF000
20 IF PEEK(&HFFFE)=252 THEN PRINT "Dies ist ein AT oder 386er PC"
30 ELSE PRINT "Dies ist ein PC"
```

Im BIOS Ihres Personalcomputers stecken aber noch mehr Informationen. Wenn Sie beispielsweise wissen möchten, welches das letzte Revisionsdatum Ihres BIOS war, lesen Sie den Speicherbereich von F000:FFF5 bis F000:FFFChex aus. Dort ist das Revisionsdatum als ASCII-Text abgelegt. Mit einem kurzen Basic-Programm läßt sich das Datum sehr einfach anzeigen:

```
10 DEF SEG=&HF000
20 FOR A%=&HFFF5 TO &HFFFCh
30 PRINT CHR$(PEEK(A%));
40 NEXT A%
```

Bedenken Sie aber bitte, daß dieses Datum in der Schreibweise »Monat:Tag:Jahr« vorliegt. »01-20-88« be-

deutet also der 20. Januar 1988. Auch die Copyright-Meldung des BIOS-Herstellers läßt sich sichtbar machen. Sie befindet sich normalerweise ab der Speicheradresse F000:E020hex. Wenn Sie das vorherige Programm geringfügig verändern, bringen Sie auch diese Information auf den Bildschirm. Listing 1 zeigt ein Beispielprogramm zur Ausgabe der aufgeführten Daten.

Vielleicht wollten Sie schon immer einmal den Zeichensatz Ihres Rechners verändern. Auch hierbei hilft Ihnen das Verständnis der Arbeitsweise des BIOS. Diese Fähigkeit weist allerdings nur die CGA-Karte auf, sie funktioniert jedoch auch bei vielen CGA-Emulatoren.

In den Textmodi haben Sie leider keinen Einfluß auf das Erscheinungsbild der Textzeichen. In den Grafikmodi lassen sich aber die ASCII-Zeichen 128 bis 255 verändern. Dazu setzen Sie den Interrupt-Vektor 1Fhex auf eine Tabelle, die für jedes Zeichen acht Byte enthält. Dabei steht jedes Bit für einen Bildpunkt in der 8x8 Punkte großen Matrix eines Zeichens. Insgesamt ist diese Tabelle 128x8 = 1024 Byte lang.

Nach dem Booten von MS-DOS ist der Interrupt-Vektor 1Fhex zunächst nicht initialisiert und zeigt auf die Adresse 0000:0000hex. Das ist auch der Grund, warum nur »Unsinn« auf dem Bildschirm erscheint, wenn Sie unter GW-Basic einen Grafikmodus (Screen 1 oder Screen 2) aktivieren und ein Zeichen mit einem ASCII-Code größer als 127 verwenden (zum Beispiel Umlaute).

Um im Grafikmodus die gleichen Zeichen zu erhalten wie im Textmodus, und nicht irgendwelche undefinierten Bitmuster, müssen Sie vor dem Start von GW-Basic das Programm »grftabl.com« resident laden. Eigene Zeichensätze lassen sich am einfachsten verwirklichen, wenn Sie zunächst »grftabl« laden und dann die resident geladene Tabelle durch Poke-Befehle verändern.

Um die Startadresse der Grafikzeichensatz-Tabelle zu finden, lesen Sie den Interrupt-Vektor 1Fhex aus. Listing 2 zeigt, wie Sie diese Adresse bestimmen. Die Adresse der Zeichensatztabelle hängt von der Systemkonfiguration des Computers ab, daher muß sie nach jeder Installation von »grftabl.com« neu ermittelt werden.

Leider sind die Grafikdaten für die ASCII-Zeichen kleiner als 128 im BIOS fest definiert und lassen sich nicht ändern. Das liegt daran, daß im BIOS die Adresse für diese Zeichensatztabelle nicht in einer Variablen im BIOS-Datensegment enthalten, sondern als absolute Adresse im BIOS-EPROM eingebrannt ist.

(Martin Hepp/ma)

Das BIOS-Datensegment 0040hex		
Adresse (hex)	Inhalt	
0000 - 0001	Portadresse der ersten seriellen Schnittstelle (»com1«)	
0002 - 0003	Portadresse der zweiten seriellen Schnittstelle (»com2«)	
0004 - 0005	Portadresse der dritten seriellen Schnittstelle (»com3«)	
0006 - 0007	Portadresse der vierten seriellen Schnittstelle (»com4«)	
0008 - 0009	Portadresse der ersten parallelen Schnittstelle (»lpt1«)	
000A - 000B	Portadresse der zweiten parallelen Schnittstelle (»lpt2«)	
000C - 000D	Portadresse der dritten parallelen Schnittstelle (»lpt3«)	
000E - 000F	Portadresse der vierten parallelen Schnittstelle (»lpt4«)	
0010	Computersystem-Statusbyte	
	Bit 0 = 0: Kein Diskettenlaufwerk installiert	
	Bit 0 = 1: Mindestens ein Diskettenlaufwerk installiert	
	Bit 1 = 0: Arithmetikprozessor 8087 nicht vorhanden	
	Bit 1 = 1: Arithmetikprozessor 8087 vorhanden	
	Bit 2 und Bit 3: Speicher auf der Hauptplatine in 16-KByte-Einheiten (bei modernen PCs mit mehr als 64 KByte auf der Hauptplatine sind beide Bits gesetzt)	
	Bit 5 = 0 und Bit 4 = 1: 40x25 Zeichen (Farbgrafik)	
	Bit 5 = 1 und Bit 4 = 0: 80x25 Zeichen (Farbgrafik)	
	Bit 5 = 1 und Bit 4 = 1: 80x25 Zeichen (Monochromgrafik)	
	Bit 6 und Bit 7: Anzahl der über Eins hinausgehenden Diskettenlaufwerke	

Tabelle 2. Das BIOS-Datensegment 0040hex von Adresse 0040:0000 bis 0040:00FF enthält die Systemvariablen und Parameter des BIOS

Das BIOS-Datensegment 0040hex			
Adresse (hex) Inhalt		Adresse (hex) Inhalt	
0011	Schnittstellen-Statusbyte Bit 0 = Unbenutzt Bit 1 und 2: Anzahl der initialisierten seriellen Schnittstellen Bit 3 = 0 (Konstant) Bit 4 = 0: Kein Gameportadapter vorhanden Bit 4 = 1: Gameportadapter vorhanden Bit 5: Unbenutzt Bit 6 und 7: Anzahl der angeschlossenen Drucker	0065	Inhalt des Modus-Auswahlregisters (Portadresse 3B8 oder 3D8hex)
0012	Testflag für Initialisierung	0066	Inhalt des Farb-Auswahlregisters (Portadresse 3D9hex)
0013 – 0014	Speicherkapazität des freien Arbeitsspeichers in KByte	0067 – 006B	früher Pufferspeicher für Kassettentfunktionen, heute für Sonderfunktionen reserviert
0015 – 0016	Reserviert	006C – 006F	32-Bit-Zähler der Systemuhr im PC (Takt: 18,2 Hz)
0017	1. Tastatur-Statusbyte Bit 0 = 1: [Right Shift] aktiv Bit 1 = 1: [Left Shift] aktiv Bit 2 = 1: [Ctrl] aktiv Bit 3 = 1: [Alt] aktiv Bit 4 = 1: [Scroll Lock] aktiv Bit 5 = 1: [Num Lock] aktiv Bit 6 = 1: [Caps Lock] aktiv Bit 7 = 1: [Ins] aktiv	0070	= 0: Seit dem Start der Systemuhr sind noch keine 24 Stunden vergangen < > 0: Seit dem Start der Systemuhr sind mehr als 24 Stunden vergangen
0018	2. Tastatur-Statusbyte Bit 0 = 0: (konstant) Bit 1 = 0: (konstant) Bit 2 = 0: (konstant) Bit 3 = 1: [Ctrl-Scroll Lock] aktiv Bit 4 = 1: [Scroll Lock] gedrückt Bit 5 = 1: [Num Lock] gedrückt Bit 6 = 1: [Caps Lock] gedrückt Bit 7 = 1: [Ins] gedrückt	0071	Bit 7 = 1: Abbruch wurde durch [Ctrl-C] oder [Ctrl-Break] herbeigeführt
0019	Aktuelles Zeichen, das mit der Alt-Taste eingegeben wurde	0072 – 0073	Wenn der Inhalt 1234hex beträgt, führt ein Reset durch [Ctrl-Alt-Del] zu einem Warmstart (kein RAM-Test)
001A	Offsetadresse des nächsten zu lesenden Zeichens im Tastaturpuffer (erforderliche Segmentadresse: 0040hex)	0074 – 0077	Diese vier Byte nehmen den Status der Festplatte(n) auf
001C	Offsetadresse der nächsten zu beschreibenden Adresse im Tastaturpuffer (erforderliche Segmentadresse: 0040hex)	0078 – 007B	Vier Timeout-Zähler für die vier parallelen Schnittstellen »lpt1« bis »lpt4« (Der Timeout-Zähler gibt in Einheiten zu einer halben Sekunde die Zeit an, die der Computer auf eine Antwort des Peripheriegerätes wartet)
001E – 003D	Tastaturpuffer (16 x 2 Byte)	007C – 007F	Vier Timeout-Zähler für die vier seriellen Schnittstellen »com1« bis »com4« (Der Timeout-Zähler gibt in Einheiten zu einer halben Sekunde die Zeit an, die der Computer auf eine Antwort des Peripheriegerätes wartet)
003E	Diskettenlaufwerke: Seek-Status	0080 – 0081	Startadresse des Tastaturpuffers als Offset relativ zum BIOS-Datensegment 0040hex
003F	Diskettenlaufwerke: Motor-Status, gibt aktive Laufwerksmotoren an	0082 – 0083	Endadresse des Tastaturpuffers als Offset relativ zum BIOS-Datensegment 0040hex
0040	Schleifenwert für Abschaltung der Diskettenlaufwerk-Motoren	0088	Monitortyp 248: CGA-Monitor (handelsüblicher Monitor) 249: EGA-Monitor 251: Monochrom-Monitor
0041	Statuswert der letzten Disketten-Operation	00A8	Grafikkarten-Verfügbarkeit 0: Keine EGA- oder VGA-Karte vorhanden 1 bis 255: EGA- oder VGA-Karte vorhanden
0042 – 0048	Speicherbereich für den Disketten-Controller 765	0090 - 004EF	Reserviert
0049	aktueller Videomodus	00F0 - 004FF	Von MS-DOS für den Anwender reserviert
004A – 004B	Bildschirmbreite in Textspalten	0100	Statusbyte für Hardcopy-Funktion 0: Hardcopy korrekt beendet 1: Harcopy in Bearbeitung 255: Fehler aufgetreten
004C – 004D	Speicherbedarf einer Bildschirmseite in Bytes	0104	Statusbyte für Einzellaufwerkmodus
004E – 004F	Offsetadresse der aktiven Bildschirmseite relativ zum Anfang des Bildschirmspeichers	0110 – 0111	Adresse des Datensegments von GW-Basic
0050 – 005F	Zwischenspeicher für die Cursorpositionen (Spalte/Zeile) auf den Bildschirmseiten 0 bis 7	0112 – 0113	Segmentadresse für Systemuhr-Interruptroutine (GW-Basic)
0060	Startzeile des Cursors (0 bis 7 (CGA), 0 bis 14 (MDA, Hercules, EGA) oder 0 bis 16 (VGA))	0114 – 0115	Offsetadresse für Systemuhr-Interruptroutine (GW-Basic)
0061	Endzeile des Cursors (0 bis 7 (CGA), 0 bis 14 (MDA, Hercules, EGA) oder 0 bis 16 (VGA))	0116 – 0117	Segmentadresse für Breakasten-Interruptroutine (GW-Basic)
0062	Nummer der aktiven Bildschirmseite	0118 – 0119	Offsetadresse für Breakasten-Interruptroutine (GW-Basic)
0063 – 0064	Portadresse des Video-Controllers der aktiven Bildschirmkarte	011A – 011B	Segmentadresse für Diskettenfehler-Serviceroutine (GW-Basic)
		011C – 011D	Offsetadresse für Diskettenfehler-Serviceroutine (GW-Basic)
		011E – 011F	Reserviert
		Bei allen Adressen ist an der ersten Speicherstelle das niederwertige und an der zweiten Stelle das höherwertige Byte abgelegt	

Tabelle 2. Das BIOS-Datensegment 0040hex von Adresse 0040:0000 bis 0040:00FF enthält die Systemvariablen und Parameter des BIOS (Fortsetzung)

```
[UT5] 10 'BIOS Detektiv »detektiv.bas«
[P75] 20 'Copyright DOS International
[YE3] 30 'Autor: Martin Hepp
[658] 40 'Dieses kurze Programm zeigt einige Informationen
[0Y7] 50 'an, die im BIOS des Computers enthalten sind.
[AY4] 60 SCREEN 0: WIDTH 80:CLS:KEY OFF
[1N6] 70 PRINT CHR$(13);"Dieser Computer ist ein ";
[Y82] 80 DEF SEG=&HFF0000
[8C3] 90 RTYPE%=PEEK(&HFFFE)
[G5A] 100 IF RTYPE%=252 THEN PRINT "IBM AT, kompatibler
AT oder 386er-PC"
[4B8] 110 IF RTYPE%=254 THEN PRINT "IBM XT oder Kompatibler"
[S98] 120 IF RTYPE%=255 THEN PRINT "IBM PC oder Kompatibler"
[X4A] 130 IF (RTYPE%<254 AND RTYPE%>252) THEN PRINT "kompatibler PC/XT"
[GV7] 140 PRINT CHR$(13);"Das BIOS ist datiert auf den ";
[851] 150 BD$=""
[W64] 160 FOR A%=&HFF5 TO &HFF5+7
[PL3] 170 BD$=BD$+CHR$(PEEK(A%))
[0N1] 180 NEXT A%
[1X7] 190 PRINT MID$(BD$,4,2);",";LEFT$(BD$,2);".19";RIGHT$(BD$,2)
[1U5] 200 PRINT CHR$(13);"BIOS-Copyright: ";
[5N2] 210 DEF SEG=&HFE00
[DP3] 220 FOR A%=&H20 TO &H20 + 56
[UA6] 230 IF PEEK(A%)>31 THEN PRINT CHR$(PEEK(A%));
[PM1] 240 NEXT A%
```

```
[1V3] 250 WHILE INKEY$<>"" :WEND
[HG7] 260 LOCATE 25,1:PRINT "Beliebige Taste für Ende";
[3P3] 270 WHILE INKEY$="" :WEND
[DU4] 280 LOCATE 25,1:PRINT SPACE$(40);
[GU2] 290 LOCATE 20,1:END
```

Listing 1. Das Programm »detektiv.bas« liefert verschiedene Informationen über Ihren Computer

```
[AR7] 10 'Grafik-Zeichensatz-Finder »graffind.bas«
[P75] 20 'Copyright DOS International
[YE3] 30 'Autor: Martin Hepp
[6Y4] 40 SCREEN 0: WIDTH 80:CLS:KEY OFF
[0HA] 50 PRINT "Die Daten für den Grafikzeichensatz der ASCII-Codes";
[AR8] 60 PRINT " 128 bis 255";CHR$(13);CHR$(13);"liegen bei Adresse ";
[AW1] 70 DEF SEG=&H0
[425] 80 SEG$=HEX$(PEEK(126)+256*PEEK(127))
[UC2] 90 L%=4-LEN(SEG$)
[9P3] 100 SS=LEFT$( "0000",L%)+SEG$
[3A5] 110 OFS$=HEX$(PEEK(124)+256*PEEK(125))
[KG2] 120 L%=4-LEN(OFS$)
[NR3] 130 OS=LEFT$( "0000",L%)+OFS$
[RT2] 140 PRINT SS;" ";OS
[ZU3] 150 WHILE INKEY$<>"" :WEND
[FG7] 160 LOCATE 25,1:PRINT "Beliebige Taste für Ende";
[1P3] 170 WHILE INKEY$="" :WEND
[LP6] 180 LOCATE 25,1:PRINT SPC(40);:LOCATE 20,1:END
```

Listing 2. Die Anfangsadresse des Grafikzeichensatzes erfahren Sie mit »graffind.bas«



DOS EXTRA Nr. 1: Ausverkauft

DOS EXTRA Nr. 2: Nicht mit Gold aufzuwiegen

**EXTRA
EXTRA
EXTRA
EXTRA
EXTRA**

DOS-Commander: Fensterorientierte Benutzeroberfläche für MS-DOS
Seitenweise Utilities, die den Umgang mit dem PC erleichtern.
Mit dem SForth-Compiler 3.1 programmieren Sie in einer neuen Dimension.
Action, Spaß und Spannung: Superspiele zum Abtippen
3D-CAD-Programm der Superlative.

Best.-Nr. 354

18,- DM*

DOS EXTRA Nr. 3: Für glasklare Informationen

**EXTRA
EXTRA
EXTRA
EXTRA
EXTRA
EXTRA**

Der große DOS-Kurs »Der PC – glasklar« in überarbeiteter Form
Datenfernübertragungs-Programm in 100% Assemblercode
Zeichensatzeditor für Epson-kompatible Drucker
Sprachdigitalisierung mit dem PC
Hercules-Grafik-Toolbox in Turbo Pascal
Grafikfunktionen für Turbo C

Best.-Nr. 356

18,- DM*



DOS EXTRA Nr 4:

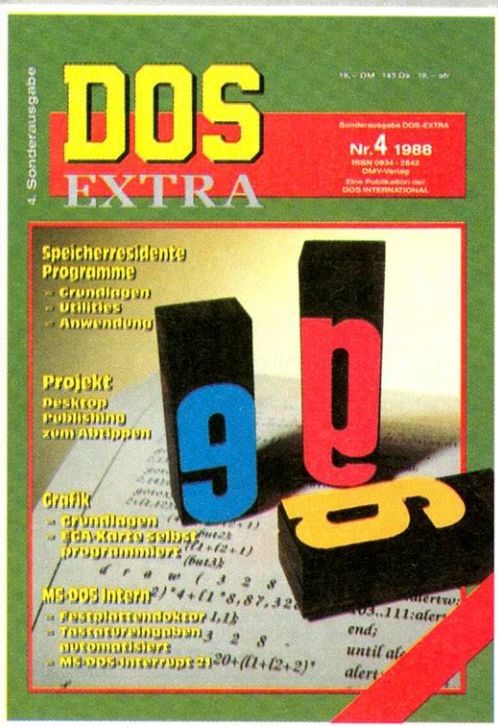
Was Sie schon immer interessiert hat

**EXTRA
EXTRA
EXTRA
EXTRA**

Desktop Publishing selbstgemacht mit SDTP
Speicherresidente Programme schreiben, analysieren und entfernen
Alles über MS-DOS-Interrupt 21 hex
Grundlagen zu Grafikkarten

Best.-Nr. 381

18,- DM*



Jetzt überall erhältlich

Die Ausgaben DOS EXTRA Nr. 2 und DOS EXTRA Nr. 3 sind nur noch über den Verlag zu beziehen. Zu allen Ausgaben ist jedoch die Databox im Format 5 1/4" oder 3 1/2" MS-DOS erhältlich. Bitte benutzen Sie die Bestellkarte.

* Unabhängig von der Anzahl der bestellten Zeitschriften berechnen wir für das Inland 3,- DM bzw. für das Ausland 5,- DM Porto und Verpackung.

DMV · Postfach 250 · 3440 Eschwege

Klare Sicht mit dem Video-Interrupt 10hex

Der Software-Interrupt mit der Nummer 10hex ist für den Assembler-Programmierer das Tor zum Bildschirm. Er gehört zu den vielfältigsten Interrupts des BIOS. Die Funktionen des Video-Interrupts erlauben den Zugriff auf alle Bildschirmkarten, vom Monochromadapter bis hin zur VGA-Karte.

Für einen Assembler-Programmierer gibt es zwei Möglichkeiten, um die Bildschirmausgabe zu beeinflussen. Als erstes kann er die Ausgabefunktionen des MS-DOS-Interrupt 21 benutzen. Die Möglichkeiten dieses Interrupts sind jedoch recht begrenzt, weil alle Funktionen zeichenorientiert arbeiten, so daß sich nur die Zeichen des ASCII-Zeichensatzes ausgeben lassen.

Weitaus leistungsfähiger und vielfältiger in ihren Anwendungsmöglichkeiten sind die Funktionen des Vi-

deo-Interrupt 10. Diese Funktionen können auf einer geeigneten Bildschirmkarte Text und Grafik ausgeben, Bildschirmbereiche auf- oder abwärts scrollen sowie einzelne Bildschirmpositionen auf Textzeichen oder Grafikpunkte abfragen. In Zusammenhang mit einer EGA- oder VGA-Karte sind die Routinen für selbstdefinierte Zeichen erwähnenswert.

Tabelle 1 zeigt alle Funktionen des Video-Interrupt 10 mit den erforderlichen Ein- und Ausgabe-Parametern.

Der Aufbau des Videomodus, dessen Angabe bei einigen Funktionen erforderlich ist, wird in Tabelle 2 dargestellt. Für den Monochromadapter und die Hercules-Grafikkarte gilt standardmäßig der Videomodus 07.

Der Aufbau des Monochrom-Attributs bei einer monochromen Grafikkarte ist in Tabelle 3 und die Farbverteilung der CGA-Farbgrafikkarte in Tabelle 4 aufgeführt. Das Farbattribut wird mit den sogenannten Bit-Planes erzeugt, deren Aufbau Sie der Tabelle 5 entnehmen können. Tabelle 6 gibt im Zusammenhang mit Funktion 1Ahex Aufschluß über die Bildschirmkarten- und Gerätekonfiguration.

Zum Monochrom- und Farbattribut ist noch zu sagen, daß es sich im Bildschirmspeicher jeweils direkt hinter dem zugehörigen Zeichen befindet, so daß jedes Zeichen im Textmodus der Grafikkarte zwei Byte an Speicherplatz belegt.

(Markus Zietlow/ma)

Die Funktionen des Video-Interrupt 10hex			
AH-Register (hex)	Eingabe	Ausgabe	Bedeutung
00	AL = Videomodus (siehe Tabelle 2)	–	Definiere den Videomodus (Text- oder Grafikdarstellung)
01	CH = 000xxxxx (xxxxx = Startresterzeile von 0 bis 1Fhex) CL = 000xxxxx (xxxxx = Endresterzeile von 0 bis 1Fhex)	–	Definiere Cursorgröße
02	BH = Bildschirmseite (0 bis 7) (BH = 0 für Grafikmodus) DH = Zeile (0 bis 24) DL = Spalte (0 bis 79)	–	Positioniere Cursor (0,0 = oben links, 39/79,24 = unten rechts)
03	BH = Bildschirmseite (0 bis 7) (BH = 0 für Grafikmodus)	CH = Cursor-Startresterzeile CL = Cursor-Endresterzeile DH = Zeile DL = Spalte	Lese Cursorposition (0,0 = oben links, 39/79,24 = unten rechts)
04	–	AH = 0: Lichtgriffel nicht ausgelöst AH = 1: Lichtgriffel ausgelöst BX = Grafikspalte (0 bis 639) CH = Grafikzeile (0 bis 199) (CX = Grafikzeile ab EGA-Karte) DH = Zeile des Lichtgriffels (0 bis 24) DL = Spalte des Lichtgriffels (0 bis 79)	Lese Lichtgriffel-Position (0,0 = oben links, 39/79,24 = unten rechts)
05	AL = Bildschirmseite (0 bis 7)	–	Wähle aktive Bildschirmseite
06	AL = Anzahl der Zeilen (frei werdende Zeilen am Fensterende werden gelöscht) (AL = 0: Lösche ganzes Fenster) BH = Attribut für die gelöschten Zeilen CH = obere Zeile (0 bis 24) CL = linke Spalte (0 bis 79) DH = untere Zeile (0 bis 24) DL = rechte Spalte (0 bis 79)	–	Scrolle Bildschirm aufwärts
07	AL = Anzahl der Zeilen (frei werdende Zeilen am Fensteranfang werden gelöscht) (AL = 0: Lösche ganzes Fenster) BH = Attribut für die gelöschten Zeilen CH = obere Zeile (0 bis 24) CL = linke Spalte (0 bis 79) DH = untere Zeile (0 bis 24) DL = rechte Spalte (0 bis 79)	–	Scrolle Bildschirm abwärts
08	BH = Bildschirmseite (0 bis 7) (BH = 0 für Grafikmodus)	AL = ASCII-Zeichen AH = Monochrom- oder Farbattribut	Lese Zeichen und Attribut von der Cursorposition
09	AL = Zeichen BH = Bildschirmseite (0 bis 7) (BH = 0 für Grafikmodus) BL = Monochrom- oder Farbattribut CX = Anzahl der Zeichen	–	Schreibe Zeichen und Attribut auf die Cursorposition (die Cursorposition ändert sich nicht)

AH-Register (hex)	Eingabe	Ausgabe	Bedeutung
0A	AL = Zeichen BH = Bildschirmseite (0 bis 7) (BH = 0 für Grafikmodus) CX = Anzahl der Zeichen	–	Schreibe Zeichen auf Cursorposition (die Cursorposition ändert sich nicht)
0B	Grafikmodus: BH = 0: BL enthält Farbe des Bildschirmrahmens und des Hintergrunds (0 bis 15) BH = 1: BL enthält Farbpalette BL = 0 für Grün, Rot und Braun BL = 1 für Kobaltblau, Violett und Weiß Textmodus: BH = 0: BL enthält Farbe des Bildschirmrahmens BL = Rahmenfarbe (0 bis 0Fhex: normal, 10 bis 1Fhex: intensiv)	–	Wähle Farbpalette oder Farbe
0C	AL = Farbe (0 bis 15) (Bit 7 = 1: XOR-Modus) BH = Bildschirmseite (ab EGA-Karte) CX = Spalte (0 bis 639) DX = Zeile (0 bis 349)	–	Setze Grafikpunkt (Pixel)
0D	BH = Bildschirm (ab EGA-Karte) CX = Spalte (0 bis 639) DX = Zeile (0 bis 349)	AL = Farbe des Punktes (0 bis 15)	Lese die Farbe eines Grafikpunktes
0E	AL = ASCII-Zeichen BH = Bildschirmseite (0 bis 7) (BH = 0 für Grafikmodus) BL = Vordergrundfarbe (im Grafikmodus)	–	Schreibe Zeichen auf Bildschirm und bewege Cursor vorwärts (Teletype-Emulation)
0F	–	AH = Zeichen pro Zeile AL = Videomodus (siehe Tabelle 2) BH = Bildschirmseite (BH = 0: Grafikmodus oder Modus 7)	Ermittle aktuellen Videomodus
10	AL = Unterfunktionsnummer (hex)	AL = Videomodus (siehe Tabelle 2) AH = Zeichen pro Zeile BH = aktive Bildschirmseite (siehe oben)	Setze Register für die Farbpalette (ab EGA-Karte)
	AL = 00 BH = Farbwert BL = zu setzendes Palettenregister		Setze ein einzelnes Paletten-Register
	AL = 01 BH = Farbwert	(siehe oben)	Setze Overscan-Register (Border)
	AL = 02 ES:DX = Adresse einer 17-Byte-Paletten-Tabelle (Byte 0 bis 15 der Tabelle enthalten die Palettenwerte, Byte 16 den Overscan-Wert)	(siehe oben)	Setze alle Paletten- und Overscan-Register
	AL = 03 BL = 0: intensive Darstellung BL = 1: blinkende Darstellung	(siehe oben)	Wechsel zwischen intensiver und blinkender Darstellung
11	AL = Unterfunktionsnummer (hex)	CX = aktuelle Zeichenhöhe DL = Zeilen pro Bildschirmseite	Charakter-Generator-Routine (ab EGA-Karte)
	AL = 00 BH = Anzahl der benötigten Bytes pro Zeichen (0 bis 31) BL = Block für den Zeichensatz (0 bis 1) CX = Anzahl der Zeichen im Zeichensatz (1 bis 256) DX = Zeichensatz-Offset (0 bis 512) ES:BP = Zeiger auf den Anfang des eigenen Zeichensatzes	–	Laden eines eigenen Zeichensatzes und Setzen eines neuen Videomodus
	AL = 01 BL = Block für den Zeichensatz (0 bis 1)	–	Laden des ROM-9x14-Monochrom-Zeichensatzes und Setzen eines neuen Videomodus
	AL = 02 BL = Block für den Zeichensatz (0 bis 1)	–	Laden des ROM-8x8-Double-Dot-Zeichensatzes und Setzen eines neuen Videomodus
	AL = 03 BL = 0: Attributbyte wird nicht als Schalter genutzt BL = 1: Attributbyte wird als Schalter genutzt	–	Wechsel zwischen zwei Zeichensätzen (Block 0 und 1) und Setzen eines neuen Videomodus
	AL = 10 BH = Anzahl der benötigten Bytes pro Zeichen (0 bis 31) BL = Block für den Zeichensatz (0 bis 1) CX = Anzahl der Zeichen im Zeichensatz (1 bis 256) DX = Zeichensatz-Offset (0 bis 512) ES:BP = Zeiger auf den Anfang des eigenen Zeichensatzes	–	Laden eines eigenen Zeichensatzes
	AL = 11 BL = Block für den Zeichensatz (0 bis 1)	–	Laden des ROM-9x14-Monochrom-Zeichensatzes
	AL = 12 BL = Block für den Zeichensatz (0 bis 1)	–	Laden des ROM-8x8-Double-Dot-Zeichensatzes
	AL = 20 ES:BP = Zeiger auf den Anfang der Tabelle mit den Grafikzeichen 128 bis 255	–	Laden der Grafikzeichentabelle
	AL = 21	–	Laden eines eigenen Grafikzeichensatzes

AH-Register (hex)	Eingabe	Ausgabe	Bedeutung
	BL = Anzahl der Zeilen einer Bildschirmseite BL = 0: Anzahl der Zeilen befindet sich in DL BL = 1:14 Zeilen pro Bildschirmseite BL = 2:25 Zeilen pro Bildschirmseite BL = 3:43 Zeilen pro Bildschirmseite CX = Anzahl der Bytes pro Zeichen (DL = Anzahl der Zeilen pro Bildschirmseite, wenn BL = 0) ES:BP = Zeiger auf den Anfang des eigenen Zeichensatzes AL = 22 BL = Anzahl der Zeilen pro Bildschirmseite (wie unter AL = 33 beschrieben) AL = 23 BL = Anzahl der Zeilen pro Bildschirmseite (wie unter AL = 33 beschrieben) CX = aktuelle Zeichenhöhe AL = 30 BL = Zeichensatzzeiger BL = 0: INT-1F-Zeiger BL = 1: INT-43-Zeiger BL = 2: ROM-8x14-Zeiger BL = 3: ROM-Double-Dot-Zeiger BL = 4: ROM-Double-Dot-Zeiger (Top) BL = 5: ROM-9x14-Zeiger (Monochrom)	- - CX = aktuelle Zeichenhöhe DL = Anzahl der Zeilen pro Bildschirmseite ES:BP = Zeiger (je nach Inhalt von BL vor dem Aufruf)	Laden des ROM-8x14-Zeichensatzes Laden des ROM-8x8-Double-Dot-Zeichensatzes Ermittle Informationen über einen Zeichensatz-zeiger
12	BL = Unterfunktionsnummer (hex) BL = 10 BL = 20	- BL = Speichergröße der Grafikkarte BL = 0: 64 KByte Bildschirmspeicher BL = 1: 128 KByte Bildschirmspeicher BL = 10: 192 KByte Bildschirmspeicher BL = 11: 256 KByte Bildschirmspeicher BH = Monitortyp BH = 0: Farbmonitor BH = 1: Monochrom-Monitor -	Ermittle Status oder aktiviere Hardcopy-Routine (ab EGA-Karte) Lies Statusinformationen Aktiviere alternative Hardcopy-Routine für Zeichenhöhen ungleich 8 oder 14
13	AL = Form der Zeichenkette AL = 0: Zeichenkette schreiben und Cursorposition nicht verändern AL = 1: Zeichenkette schreiben und Cursor mitführen (Teletype-Emulation) AL = 2: Zeichenkette sowie Attributbyte schreiben und Cursorposition nicht verändern AL = 3: Zeichenkette sowie Attributbyte schreiben und Cursor mitführen (Teletype-Emulation) BH = Bildschirmseite BL = Attributbyte (bei AL = 0 oder AL = 1) CX = Länge der Zeichenkette DH = Zeile der Startposition DL = Spalte der Startposition ES:BX = Adresse der Zeichenkette	-	Gebe eine Zeichenkette auf den Bildschirm aus (ähnlich Funktion 9, aber nur für AT)
14	-	-	Reserviert
15	-	-	Reserviert
16	-	-	Reserviert
17	-	-	Reserviert
18	-	-	Reserviert
19	-	-	Reserviert
1A	AL = Unterfunktionsnummer AL = 0 AL = 1	- AL = 1A BH = nicht aktiver Monitor BL = aktiver Monitor AL = 1A BH = nicht aktiver Monitor BL = aktiver Monitor	Videokonfiguration (VGA-Karte) Ermittle die Videokonfiguration (siehe Tabelle 6) Setze die Videokonfiguration (siehe Tabelle 6)
1B	BX = 0 ES:DI = Adresse eines 64-Byte-Puffers	AL = 1B ES:DI = Puffer, der die gewünschten Daten enthält	Ermittle den Video- und BIOS-Status
1C	AL = Unterfunktionsnummer AL = 0: CX = Status Bit 0 = 1: Video-Hardware-Status Bit 1 = 1: Video-BIOS-Datenbereiche Bit 2 = 1: Video-DAC-Status Bit 3 bis 15: Reserviert AL = 1 CX = Status (wie in Unterfunktion 0 beschrieben) ES:BX = Adresse des Puffers AL = 2 CX = Status (wie in Unterfunktion 0 beschrieben) ES:BX = Adresse des Puffers	- AL = 1C BX = Puffergröße in 64-Byte-Blöcken - -	Ändere, lies oder setze den Video-Status Lies die Größe des Video-Puffers Kopiere den Status in den angegebenen Datenblock Setze den Video-Status auf die im Puffer abgespeicherten Werte

Die Text- und Grafikkordinatenwerte der Funktionen sind in dezimaler Schreibweise angegeben

Tabelle 1. Die Funktionen des Video-Interrupt 10hex sind zum Teil von der verwendeten Grafikkarte abhängig

Der Videomodus für die Funktionen 00, 0F, 10 und 11 des Video-Interrupt 10						
AL (hex)	Modus	Grafikauflösung	Textauflösung	Farben	Bildschirmseiten	Anmerkung
00	Text	–	40x25	16 Graustufen	8	–
01	Text	–	40x25	16	8	–
02	Text	–	80x25	16 Graustufen	4	Standardwert für CGA-, EGA- und VGA-Karte
03	Text	–	80x25	16	4	–
04	Grafik	320x200	–	4	1	–
05	Grafik	320x200	–	4 Graustufen	1	–
06	Grafik	640x200	–	monochrom	1	–
07	Text	–	80x25	monochrom	1	Standardwert für Monochromadapter und Hercules-Grafikkarte
08	Grafik	160x200	–	16	2	nur auf IBM PC Junior
09	Grafik	320x200	–	16	1	nur auf IBM PC Junior
0A	Grafik	640x200	–	4	2	nur auf IBM PC Junior
0B						Reserviert
0C						Reserviert
0D	Grafik	320x200	–	16	4	nur EGA- und VGA-Karte
0E	Grafik	640x200	–	16	2	nur EGA- und VGA-Karte
0F	Grafik	640x350	–	monochrom	1	nur EGA- und VGA-Karte
10	Grafik	640x350	–	16	1	nur EGA- und VGA-Karte
11	Grafik	640x480	–	2	1	nur VGA-Karte
12	Grafik	640x480	–	16	1	nur VGA-Karte
13	Grafik	320x200	–	256	1	nur VGA-Karte

Tabelle 2. Der Videomodus bestimmt die Anzahl der Farben sowie die Text- und Grafikauflösung

Das Attribut des Monochromadapters		
Bit-Kombination	dezimaler Wert	Darstellung
00000000	0	Unsichtbar
00000001	1	Unterstrichen
00000111	7	Normal
00001001	9	Leuchtend und unterstrichen
00001111	15	Leuchtend
01110000	112	Invers
10000001	129	Blinkend und unterstrichen
10000111	135	Blinkend
10001001	137	Blinkend, leuchtend und unterstrichen
10001111	143	Blinkend und leuchtend
11110000	240	Blinkend und invers

Tabelle 3. Der Monochromadapter und die Hercules-Grafikkarte können Textzeichen auf verschiedene Arten darstellen

Die Aufteilung des Bildschirmspeichers in Bit-Planes				
Plane	Plane 0	Plane 1	Plane 2	Plane 3
Farbe	Blau	Grün	Rot	Intensiv

Tabelle 5. Diese vier Bit-Planes bestimmen bei der CGA-Grafikkarte die Farbdarstellung

Die Farben der CGA-Grafikkarte					
Nummer	Farbe	Blau	Grün	Rot	Intensiv
00	Schwarz	0	0	0	0
01	Blau	1	0	0	0
02	Grün	0	1	0	0
03	Kobaltblau	1	1	0	0
04	Rot	0	0	1	0
05	Violett	1	0	1	0
06	Braun	0	1	1	0
07	Weiß	1	1	1	0
08	Grau	0	0	0	1
09	Hellblau	1	0	0	1
0A	Hellgrün	0	1	0	1
0B	Hellkobaltblau	1	1	0	1
0C	Hellrot	0	0	1	1
0D	Hellviolett	1	0	1	1
0E	Gelb	0	1	1	1
0F	Hellweiß	1	1	1	1

Tabelle 4. 16 verschiedene Farben lassen sich mit der CGA-Grafikkarte erzeugen

Die Videokonfiguration für die Funktion 1Ahex			
Wert (hex)	Konfiguration	Wert (hex)	Konfiguration
00	Kein Monitor vorhanden	06	Professioneller Grafik-Controller
01	Monochromadapter mit Monochrom-Monitor	07	VGA-Karte mit analogem Monochrom-Monitor
02	CGA-Farbgrafikkarte mit Farbmonitor	08	VGA-Karte mit analogem Farbmonitor
03	Reserviert	0A	MCGA-Karte mit digitalem Farbmonitor
04	EGA-Karte mit Farbmonitor	0B	MCGA-Karte mit analogem Monochrom-Monitor
05	EGA-Karte mit Monochrom-Monitor	0C	MCGA-Karte mit analogem Farbmonitor

Tabelle 6. Die Funktion 1Ahex arbeitet nur mit einer VGA-Karte

Der Disketten- und Festplatten-Interrupt 13hex

Über den Interrupt 13hex greifen Sie ohne den Umweg über die Routinen von MS-DOS direkt auf eine Diskette oder Festplatte zu – nicht nur für »Kopierschutzknacker« ein interessantes Thema!

Zur Verwaltung von Dateien und Programmen bieten zweifellos die Funktionen des MS-DOS-Interrupt 21 den größeren Komfort. Doch wenn Sie beispielsweise einen ganz bestimmten Sektor von Ihrer Diskette oder Festplatte lesen möchten, wenn Sie einzelne Sektoren beschreiben oder überprüfen wollen, wenn Sie spurweise formatieren oder defekte Sektoren kennzeichnen möchten, dann ist der Interrupt 13hex genau das richtige für Sie.

Tabelle 1 zeigt die wichtigsten Funktionen des Interrupt 13hex und Tabelle 2 die den Fehlercodes entsprechenden Fehlermeldungen.

Durch den Aufruf der Funktion 00 führt der Computer einen Reset im Disketten-Controller und in den angeschlossenen Diskettenlaufwerken oder der angeschlossenen Festplatte durch. Das heißt, die Funktion 00 versetzt die Geräteeinheit in den Einschaltzustand. Durch den Reset wird der Schreib-/Lesekopf auf eine festgelegte Position bewegt, und die folgenden Diskettenoperationen beginnen an dieser Spur.

Bei der Funktion 02 müssen Sie beachten, daß sich alle gelesenen Sektoren in einer Spur auf einer Seite befinden. Weil das Betriebssystem nicht überprüfen kann, wie viele Sektoren in einer Spur beziehungsweise auf einer Seite vorhanden sind, kann eine Nichtbeachtung dieser Warnung zu Fehlern führen. Das gleiche gilt für die Funktion 03.

Entgegen der Meinung vieler Anwender vergleicht die Funktion 04 die angesprochenen Sektoren nicht mit dem Inhalt eines Puffers im Arbeitsspeicher. Sie überprüft vielmehr die Spuren auf ihre Lesbarkeit, indem die Prüfsummen kontrolliert werden. Funktion 05 formatiert eine einzelne Spur auf einem Diskettenlaufwerk oder einer Festplatte. Zu diesem Zweck müssen Sie der Routine die Adresse eines Parameterblocks übergeben, der eine Kennung enthält, die jeder Sektor erhalten soll. Den Aufbau des Parameterblocks finden Sie in Tabelle 3.

Die Funktionen, die nur für einen AT gelten, beginnen mit der Nummer 08.

Die Funktionen des Disketten- und Festplatten-Interrupt 13hex				
AH-Register (hex)	Computer	Eingabe	Ausgabe	Bedeutung
00	PC/XT/AT	DL=0: Diskettenlaufwerke DL=80: erste Festplatte DL=81: zweite Festplatte, usw.	—	Initialisiere Diskettenlaufwerke oder Festplatte neu
01	PC/XT/AT	DL=0: Diskettenlaufwerke DL=80: erste Festplatte DL=81: zweite Festplatte, usw.	AL=Fehlercode	Lesen den Fehlercode der letzten Diskettenoperation (nur Festplatte: Setze Fehlercode anschließend auf 0)
02	PC/XT/AT	AL=Anzahl der Sektoren Diskettenlaufwerk: CH=Spur (0 bis 27) CL=Sektor (1 bis 9) DH=Kopf (0 oder 1) DL=Laufwerk (0 bis 3) ES:BX=Lesebuffer Festplatte: CH=Spur (0 bis 3FFhex) (Bit 6 und 7 von CL=Bit 8 und 9 von CH) Bit 0 bis 4 von CL=Sektor (0 bis 11) DH=Kopf (0 bis 7) DL=Laufwerk (80 bis 87) ES:BX=Lesebuffer	C-Flag=0: kein Fehler C-Flag=1: Fehler aufgetreten AH=Fehlercode	Lesen einen oder mehrere Sektoren von Diskette oder Festplatte
03	PC/XT/AT	AL=Anzahl der Sektoren Diskettenlaufwerk: CH=Spur (0 bis 27) CL=Sektor (1 bis 9) DH=Kopf (0 oder 1) DL=Laufwerk (0 bis 3) ES:BX=Schreibpuffer Festplatte: CH=Spur (0 bis 3FFhex) (Bit 6 und 7 von CL=Bit 8 und 9 von CH) Bit 0 bis 4 von CL=Sektor (0 bis 11) DH=Kopf (0 bis 7) DL=Laufwerk (80 bis 87) ES:BX=Schreibpuffer	C-Flag=0: kein Fehler C-Flag=1: Fehler aufgetreten AH=Fehlercode	Schreiben einen oder mehrere Sektoren auf Diskette oder Festplatte
04	PC/XT/AT	AL=Anzahl der Sektoren Diskettenlaufwerk: CH=Spur (0 bis 27) CL=Sektor (1 bis 9) DH=Kopf (0 oder 1) DL=Laufwerk (0 bis 3) Festplatte: CH=Spur (0 bis 3FFhex) (Bit 6 und 7 von CL=Bit 8 und 9 von CH) Bit 0 bis 4 von CL=Sektor (0 bis 11) DH=Kopf (0 bis 7) DL=Laufwerk (80 bis 87)	C-Flag=0: kein Fehler C-Flag=1: Fehler aufgetreten AH=Fehlercode	Überprüfen einen oder mehrere Sektoren auf Diskette oder Festplatte
05	PC/XT/AT	ES:BX=Formatpuffer Diskettenlaufwerk: CH=Spur (0 bis 27) DH=Kopf (0 oder 1) DL=Laufwerk (0 bis 3) Festplatte: CH=Spur (0 bis 3FFhex) (Bit 6 und 7 von CL=Bit 8 und 9 von CH) DH=Kopf (0 bis 7) DL=Laufwerk (80 bis 87)	C-Flag=0: kein Fehler C-Flag=1: Fehler aufgetreten AH=Fehlercode	Formatieren eine einzelne Spur auf Diskette oder Festplatte
06	XT/AT	CH=Spur (0 bis 3FF)hex (Bit 6 und 7 von CL=Bit 8 und 9 von CH) DH=Kopf (0 bis 7) DL=Laufwerk (80 bis 87)	C-Flag=0: kein Fehler C-Flag=1: Fehler aufgetreten AH=Fehlercode	Kennzeichnen defekte Festplattenspur mit einem entsprechenden Statusbit
07	XT/AT	CH=Spur (0 bis 1023) (Bit 6 und 7 von CL=Bit 8 und 9 von CH) DH=Kopf (0 bis 7) DL=Laufwerk (80 bis 87)	C-Flag=0: kein Fehler C-Flag=1: Fehler aufgetreten AH=Fehlercode	Formatieren Festplatte ab einer bestimmten Spur

AH-Register (hex)	Computer	Eingabe	Ausgabe	Bedeutung
08	AT	DL = Laufwerk (80 bis 87)	C-Flag=0: kein Fehler C-Flag=1: Fehler aufgetreten AH=Fehlercode CH=Anzahl der Spuren (Bit 6 und 7 von CL=Bit 8 und 9 von CH) CL=Anzahl der Sektoren DH=Anzahl der Köpfe DL=Anzahl der angeschlossenen Festplatten	Ermittle Laufwerksdaten der Festplatte
09	AT	—	C-Flag=0: kein Fehler C-Flag=1: Fehler aufgetreten AH=Fehlercode	Festplattenparametertabelle initialisieren
0A	AT	AL=Anzahl der Sektoren CH=Spur (0 bis 3FFhex) (Bit 6 und 7 von CL=Bit 8 und 9 von CH) Bit 0 bis 4 von CL=Sektor (0 bis 11) DH=Kopf (0 bis 7) DL=Laufwerk (80 bis 87) ES:BX=Lesepuffer	C-Flag=0: kein Fehler C-Flag=1: Fehler aufgetreten AH=Fehlercode	Lese einen oder mehrere lange Sektoren von Festplatte
0B	AT	AL=Anzahl der Sektoren CH=Spur (0 bis 3FFhex) (Bit 6 und 7 von CL=Bit 8 und 9 von CH) Bit 0 bis 4 von CL=Sektor (0 bis 11) DH=Kopf (0 bis 7) DL=Laufwerk (80 bis 87) ES:BX=Schreibpuffer	C-Flag=0: kein Fehler C-Flag=1: Fehler aufgetreten AH=Fehlercode	Schreibe einen oder mehrere lange Sektoren auf Festplatte
0C	AT	CH=Zylindernummer DH=Kopf (0 bis 7) DL=Laufwerk (80 bis 87)	C-Flag=0: kein Fehler C-Flag=1: Fehler aufgetreten AH=Fehlercode	Positioniere Schreib-/Lesekopf auf Zylinder
0D	AT	DL=Laufwerk (80 bis 87)	C-Flag=0: kein Fehler C-Flag=1: Fehler aufgetreten AH=Fehlercode	Alternativer Reset für die Festplatte (ähnlich wie Funktion 0)
10	AT	DL=Laufwerk (80 bis 87)	C-Flag=0: kein Fehler C-Flag=1: Fehler aufgetreten AH=Fehlercode	Überprüfe, ob Festplatte bereit ist
11	AT	DL=Laufwerk (80 bis 87)	C-Flag=0: kein Fehler C-Flag=1: Fehler aufgetreten AH=Fehlercode	Parke Schreib-/Leseköpfe der Festplatte auf Spur 0
13	AT	DL=Laufwerk (80 bis 87)	C-Flag=0: kein Fehler C-Flag=1: Fehler aufgetreten AH=Fehlercode	Prüfe Lesbarkeit von Sektor 0 auf jeder Festplattenspur
14	AT	DL=Laufwerk (80 bis 87)	C-Flag=0: kein Fehler C-Flag=1: Fehler aufgetreten AH=Fehlercode	Führe Selbsttest für Festplatte durch
15	AT	DL=Laufwerksnummer	AH=Laufwerkstyp AH=0: kein Laufwerk AH=1: Diskette ohne Möglichkeit, einen Diskettenwechsel zu erkennen AH=2: Diskette mit Möglichkeit, einen Diskettenwechsel zu erkennen AH=3: Festplatte (wenn AH=3, dann enthalten CX und DX die Anzahl der 512-Byte-Sektoren)	Stelle Laufwerkstyp fest
16	AT	DL=Laufwerksnummer	AH=Wechselstatus AH=0: kein Wechsel AH=1: Diskette gewechselt	Feststellen, ob Diskettenwechsel erfolgt ist
17	AT	DL=Laufwerksnummer AL=Laufwerkstyp AL=0: kein Laufwerk angeschlossen AL=1: Diskettenlaufwerk für 360-KByte-Disketten AL=3: Diskettenlaufwerk für 1,2-MByte-Disketten	—	Lege Laufwerkstyp fest

Die Register BX, CX, DX, CS, SI, DI und BP werden automatisch gesichert

Tabelle 1. Die Funktionen des Interrupt 13hex erlauben den Zugriff auf die Datenträger des PC

Die Fehlercodes des Interrupt 13		
AH-Register (hex)	(Diskette, Festplatte)	Fehlermeldung
00	D,F	Kein Fehler aufgetreten
01	D,F	Ungültiger Befehl
02	D,F	Adreßmarke des angesprochenen Sektors ist defekt
03	D	Datenträger ist schreibgeschützt
04	D,F	Sektor nicht gefunden
05	F	Erfolgloser Reset-Versuch
07	F	Fehler bei der Initialisierung
08	D	Überlauf im DMA-Controller (Datenübertragung zu schnell)
09	D,F	Ungültiger Versuch, die 64-KByte-Grenze des DMA-Controllers zu überschreiten
10	D	Ungültiger CRC (Prüfsummenfehler beim Lesen oder Verifizieren)
	F	Nicht korrigierbarer Lesefehler, Daten wurden nicht gelesen
11	F	Daten trotz Prüfsummenfehler gelesen
20	D,F	Fehler im Controller (zum Beispiel Spur 0 nicht gefunden)
40	D,F	Spur nicht gefunden
80	D,F	Laufwerk reagiert nicht (Wartezeit überschritten)
BB	F	Fehlermeldung vom BIOS (zum Beispiel falscher Interleave-Wert)
FF	F	Nicht aufschlüsselbarer Fehler aufgetreten

Tabelle 2. Der Wert des Fehlercodes im AH-Register repräsentiert eine Fehlermeldung

Der Aufbau des Formatpuffers:	
Byte	Inhalt
Byte 00	Spur für den ersten Sektor
Byte 01	Kopf für den ersten Sektor
Byte 02	Nummer für den ersten Sektor
Byte 03	Anzahl der Bytes im ersten Sektor (0=128, 1=256, 2=512, 3=1024)
Byte 04	Spur für den zweiten Sektor
Byte 05	Kopf für den zweiten Sektor
Byte 06	Nummer für den zweiten Sektor
Byte 07	Anzahl der Bytes im zweiten Sektor
Byte 08	Spur für den dritten Sektor
Byte nn	und so weiter

Tabelle 3. Der Formatpuffer enthält die Informationen für die Funktion 05

Die Funktion 09 für einen AT entspricht der Funktion 01 für den PC. Nach dem Aufruf zeigt der Interrupt 41hex auf die Parameter für Laufwerk 0 und der Interrupt 46hex auf die Parameter für Laufwerk 1.

Die Funktionen 0A und 0Bhex schreiben und lesen im Gegensatz zu den Funktionen 02 und 03 lange ECC-Sektoren. Ein langer Sektor enthält einen Fehlercode aus vier Bytes (»ECC«), der zur Fehlererkennung und -korrektur benutzt wird.

Doch bevor Sie nun anfangen, Ihre Diskette über den Interrupt 13hex zu untersuchen, eine eindringliche Warnung: Schon ein kleiner Fehler beim Programmieren kann die Daten auf Ihrer Diskette teilweise oder ganz zerstören. Das gleiche gilt natürlich auch für die Festplatte. An eine Festplatte sollten Sie sich über den Interrupt 13hex wirklich nur dann wagen, wenn Sie weitreichende Kenntnisse bezüglich des logischen Aufbaus einer Festplatte besitzen und über einige Erfahrung im Umgang mit den Funktionen des Interrupt 13hex verfügen.

(Markus Zietlow/ma)

Am Draht zur Welt:

Schnittstellen-Interrupt 14hex

Wenn Sie in Assembler-Programmen Daten übertragen wollen, kommen Sie am Interrupt 14 nicht vorbei, denn dieser Interrupt ist für die Kommunikation mit einer seriellen Schnittstelle zuständig.

Daß die Programmierung einer seriellen Schnittstelle nicht ganz einfach ist, zeigt der Interrupt 14 (Tabelle 1). Zwar ist dieser Interrupt mit nur vier Funktionen sehr übersichtlich, doch haben es die Parameter dieser Funktionen in sich.

Die Initialisierung der Schnittstelle über die Funktion 00 durch das Konfigurationsbyte (Tabelle 2 und 3) ist noch relativ leicht zu bewerkstelligen, doch die Auswertung des Zeilen- und Modem-Statusregisters (Tabelle 4 und 5) während der Daten-

übertragung ist schon etwas kniffliger. Aber für DFÜ-Experten wird dieser Signal-Dschungel sicherlich keine Hürde darstellen, und wer in der Materie nicht so sicher ist, kann auf ein fertiges Programm (zum Beispiel »Procomm Plus« aus der Public-Domain-Bibliothek) zurückgreifen. Die Datenfernübertragung ist aber nicht das einzige Anwendungsgebiet für die serielle Schnittstelle. Weil mittlerweile fast jeder Drucker mit einer seriellen Schnittstelle ausgestattet ist, kann beispielsweise auch dieses Peripheriegerät angeschlossen werden.

(Markus Zietlow/ma)

Die Funktionen des Schnittstellen-Interrupt 14hex			
AH-Register (hex)	Eingabe	Ausgabe	Bedeutung
00	AL = Konfiguration DX = Schnittstelle (0 bis 3)	AH = Zeilen-Statusregister AL = Modem-Statusregister	Initialisiere eine serielle Schnittstelle
01	AL = ASCII-Code des zu sendenden Zeichens DX = Schnittstelle (0 bis 3)	AH = Zeilen-Statusregister (Bit 7 = 0: Zeichen wurde übertragen Bit 7 = 1: Wartezeitfehler)	Sende ASCII-Zeichen an die serielle Schnittstelle
02	DX = Schnittstelle (0 bis 3)	AL = ASCII-Code des empfangenen Zeichens AH = Zeilen-Statusregister (Bit 7 = 0: Zeichen wurde übertragen Bit 7 = 1: Wartezeitfehler)	Empfange ASCII-Zeichen von der seriellen Schnittstelle
03	DX = Schnittstelle (0 bis 3)	AH = Zeilen-Statusregister AL = Modem-Statusregister	Ermittle den Schnittstellen-Status
Die Funktionen des Schnittstellen-Interrupt 14hex beeinflussen nur die Register AX, DX und die Statusbits			

Tabelle 1. Nur vier Funktionen reichen zur Programmierung der seriellen Schnittstelle aus

Das Konfigurationsbyte:	
Bit	Bedeutung
Bit 0 = 0	Sieben Datenbit
Bit 0 = 1	Acht Datenbit
Bit 1 = 1	(konstant)
Bit 2 = 0	Ein Stopbit
Bit 2 = 1	Zwei Stopbit
Bit 3 = 0	Parität gesperrt
Bit 3 = 1	Parität freigegeben
Bit 4 = 0	Ungerade Parität
Bit 4 = 1	Gerade Parität
Bit 5 bis Bit 7	Baudcode (siehe Tabelle 3)

Tabelle 2. Das Konfigurationsbyte definiert die Parameter zur Datenübertragung

Die Codierung der Baudrate im Konfigurationsbyte	
Baudcode	Baudrate
0	110
1	150
2	300
3	600
4	1200
5	2400
6	4800
7	9600

Tabelle 3. Drei Bits im Konfigurationsbyte bestimmen die Datenübertragungsgeschwindigkeit

Der Aufbau des Zeilen-Statusregisters		
Bit	Bezeichnung	Funktion
0	Data Ready (DR)	Empfängerdaten bereit
1	Overrun Error (OE)	Überlauffehler
2	Parity Error (PE)	Paritätsfehler
3	Framing Error (FE)	Rahmenfehler
4	Break Indicator (BI)	Unterbrechung erkannt
5	Transmit Holding Register Empty (THRE)	Transmit-Holding-Register unbesetzt
6	Transmit Shift Register Empty (TSRE)	Transmit-Shift-Register unbesetzt
7	Time Out Error (TOE)	Wartezeitfehler

Tabelle 4. Das Zeilen-Statusregister zeigt Fehler und Bereitschaftszustände bei der Aufteilung und dem Zusammensetzen eines Bytes an

Der Aufbau des Modem-Statusregisters		
Bit	Bezeichnung	Funktion
0	Delta Clear To Send (DCTS)	Delta-Modem bereit zum Senden
1	Delta Data Set Ready (DDSR)	Delta-Modem ist angeschaltet
2	Trailing Edge Ring Indicator (TREI)	Delta-Telefon läutet
3	Delta Data Carrier Detect (DDCD)	Delta-Verbindung zum Empfänger-Modem hergestellt
4	Clear To Send (CTS)	Modem zum Senden bereit
5	Data Set Ready (DSR)	Modem ist angeschaltet
6	Ring Indicator (RI)	Telefon läutet
7	Data Carrier Detect (DCD)	Verbindung zum Empfänger-Modem hergestellt

Tabelle 5. Das Modem-Statusregister gibt den Zustand der Schnittstellensignale an

Antiquität: Kassetten-Interrupt 15hex

Sie werden es vielleicht nicht glauben, doch die ersten IBM PCs verfügten noch über einen Kassettenrecorder-Anschluß. Für die Datenübertragung auf Kassette war der Interrupt 15hex vorgesehen, der beim AT um zusätzliche Funktionen erweitert wurde.

Der Interrupt 15 (Tabelle 1) ist in einem PC der wohl am seltensten benutzte Interrupt, denn die wenigsten PC-Besitzer werden an ihrem Computer einen Kassettenrecorder angeschlossen haben.

Im Gegensatz zu den Disketten- und Festplattenfunktionen schaltet der Computer bei Ein- und Ausgabeoperationen über den Kassettenrecorder diesen nicht automatisch ab. Wenn Sie Funktion 00 nutzen wollen, müssen Sie deshalb mit einer Zeitverzögerung arbeiten, um den Start des Motors abzuwarten, bevor der erste Schreib-/Lesezugriff erfolgt. Mit den

Funktionen 02 und 03 ist der Zugriff auf den Kassettenrecorder erheblich komfortabler, weil hier das Ein- und Ausschalten des Recorders automatisch geschieht. Außerdem enthält das AH-Register beim Beenden der beiden Funktionen ein Kassettenstatusbyte (Tabelle 2).

Auch der Amstrad PC verwendet einen modifizierten Interrupt 15hex

Ganz anders sieht die Situation bei AT- und Amstrad-PC-Besitzern aus. In diesen Computermodellen wurde

der Interrupt 15hex ganz erheblich erweitert. Beim Amstrad PC sind die Funktionen des Interrupt 15hex für die Verwaltung des batteriegepufferten RAM (NVR) und die Grafik-Bitplanes im Grafikmodus 2 zuständig.

Der AT hat mit dem Interrupt 15hex viele Funktionen erhalten, die das Betriebssystem sinnvoll ergänzen. Einige Funktionen sind aber so kompliziert, daß Sie bei Interesse auf weiterführende Literatur zurückgreifen sollten.

(Markus Zietlow/ma)

Das Kassettenstatusbyte	
AH-Register (hex)	Fehlermeldung
00	Kein Fehler aufgetreten
01	Ungültiger CRC (Prüfsummenfehler beim Lesen oder Verifizieren)
02	Keine Datenübergänge gefunden
03	Daten auf Kassette nicht gefunden
04	Kein Leitsatz (Header) gefunden
80	Ungültiger Befehl

Tabelle 2. Das Kassettenstatusbyte zeigt an, ob ein Lese- oder Schreibversuch erfolgreich war

Die Funktionen des Kassetten-Interrupt 15hex				
AH-Register (hex)	Computer	Eingabe	Ausgabe	Bedeutung
00	PC/XT	-	-	Schalte Kassettenrecordermotor ein
01	PC/XT	-	-	Schalte Kassettenrecordermotor aus
02	PC/XT	CX = Anzahl der Bytes ES:BX = Lesebuffer	AH = Kassettenstatus DX = Anzahl der gelesenen Bytes ES:BX = Adresse hinter dem zuletzt gelesenen Byte	Schalte Kassettenrecordermotor ein, lese Datenblock von Kassette und schalte Kassettenrecordermotor wieder aus
03	PC/XT/AT	CX = Anzahl der Bytes ES:BX = Schreibbuffer	AH = Kassettenstatus ES:BX = Adresse hinter dem zuletzt geschriebenen Byte	Schalte Kassettenrecordermotor ein, schreibe Datenblock auf Kassette, und schalte Kassettenrecordermotor wieder aus
80	AT	BX = Gerätenummer CX = Prozeßart	-	Öffne AT-Schnittstelle
81	AT	BX = Gerätenummer CX = Prozeßart	-	Schließe AT-Schnittstelle
82	AT	BX = Gerätenummer	-	Beende Programm
83	AT	AL = Auswahl AL = 0: Intervall setzen AL = 1: abbrechen CX = Wartezeit in Mikrosekunden (MSW) DX = Wartezeit in Mikrosekunden (LSW) ES:BX = Adresse des zu manipulierenden Byte	-	Setze nach Wartezeit Bit 7 unter Adresse ES:BX
84	AT	DX = Unterfunktionsnummer DX = 0	- C-Flag = 0: Spiele-Adapter vorhanden AL = Status der Feuerknöpfe Bit 7 = 1: erster Feuerknopf des ersten Joysticks gedrückt Bit 6 = 1: zweiter Feuerknopf des ersten Joysticks gedrückt Bit 5 = 1: erster Feuerknopf des zweiten Joysticks gedrückt Bit 4 = 1: zweiter Feuerknopf des zweiten Joysticks gedrückt C-Flag = 1: kein Spiele-Adapter angeschlossen AX = X-Position des ersten Joysticks	Joystick-Abfrage Frage momentanen Feuerknopf-Status ab
		DX = 1		Ermittle Joystick-Stellung

AH-Register (hex)	Computer	Eingabe	Ausgabe	Bedeutung
			BX = Y-Position des ersten Joysticks CX = X-Position des zweiten Joysticks DX = Y-Position des zweiten Joysticks	
85	AT	AL = Status AL = 0: Taste wurde gedrückt AL = 1: Taste wurde losge- lassen	-	Taste [SysReq] wurde gedrückt oder losgelassen
86	AT	CX = Wartezeit in Mikrosekun- den (MSW) DX = Wartezeit in Mikrosekun- den (LSW)		Warte gewisse Zeit in Mikrosekunden
87	AT	CX = Anzahl der zu verschie- benden Worte ES:SI = Zeiger auf Global-De- scriptor-Tabelle	C-Flag = 0: kein Fehler C-Flag = 1: Fehler aufgetreten AH = Fehlercode AH = 1: RAM-Paritäts-Fehler AH = 2: Global-Descriptor-Tabelle bei Funktionsaufruf nicht in Ordnung AH = 3: Protected-Mode konnte nicht rich- tig initialisiert werden	Verschiebe Speicherblock zwischen Extended Memory und Arbeitsspeicher
88	AT	-	AX = Größe des Speichers	Ermittle Speichergröße des Extended Memory in KByte
89	AT	-	-	Schalte in den Protected Mode des Prozessors um
90	AT	AL = Typencode	-	Rufe Schleife »Gerät ist beschäftigt« auf
91	AT	AL = Typencode	-	Setze Flags und beende Interrupt

Der ehemalige Kassetteninterrupt wurde beim AT erheblich erweitert

Tabelle 1. Der Kassettenrecorder-Interrupt 15hex hat diese Bezeichnung von seinen ersten vier Funktionen erhalten

ALBATROS-COMPUTER GmbH

7107 BAD WIMPFEN, MAGNUSSTR. 9, TEL. 0 70 63/70 57 (KEIN LADENVERKAUF)

➔ EIN NAME, DEN MAN SICH MERKEN SOLLTE. ➔

Computer aus eigener Fertigung. Unser Service endet nicht mit der Auslieferung. Wir bauen Ihnen jeden Computer bis hin zum 20 Mhz AT mit 16 MHz Coprozessor oder auch 24 Mhz 386. Auch in der Meß- und Betriebsdatenerfassung sind wir ein kompetenter Partner für Problemlösungen.

Zur Zeit bieten wir an:

AC 88/03, XT kompatibel mit 360 KB FDD und 20 MB HDD, 512 KB RAM (max. 640 KB on board) 4,77/10 MHz schaltbar, Mainboard mit Monochrom/Color/Graphik Controller, Multi I/O mit parallel, seriell, Mouse, Game und Echtzeituhr, Diskettencontroller für 360 KB und 1,2 MB Laufwerke, HDD Controller, 150 Watt Netzteil, AT-Look Gehäuse mit LED's und Schalter, deutsche Tastatur, MS-DOS und 12" Monitor grün Preis komplett: 2.099,- DM

AC 286/03, Baby AT-kompatibel 16 MHz Speed mit 1,2 MB FDD und 41 MB HDD, 640 KB 0 Wait (max. 4 MB on board) 8 Slots, Legal Bios, Echtzeituhr, schneller Kombicontroller für 2 FDD und 2 HDD, AT I/O Karte mit parallel und seriell Schnittstelle, monochrome Graphik-karte, 101 Tasten deutsch, 200 Watt Netzteil, Baby Gehäuse mit LED's und Schaltern, 14" Monochrome Monitor Flatscreen amber mit Drehfuß Preis komplett: 3.499,- DM

AC 386/01, 80386 CPU, 16 MHz (21 MHz Speed) mit 1,2 MB FDD und 20 MB HDD, 1 MB RAM 0 Wait (max. 8 MB on board) 8 Slots, Kombicontroller für Interleave 1:1, I/O Karte mit parallel und seriell, Monochrome/Graphik Karte mit 720 x 348 Auflösung und Printerport, 101 Tasten deutsch, 200 Watt Netzteil, Baby-Gehäuse mit LED's und Schalter, 14" Monochrome Monitor Flatscreen amber mit Drehfuß Preis komplett: 6.198,- DM

Portables mit LCD 640 x 400 XT ab 2.500,- DM, AT ab 3.400,- DM

Ab sofort auch Software im Angebot: z.B. dBFast dB-Compiler 215,- DM

Lieferung erfolgt ab Lager solange der Vorrat reicht. Händler fordern unsere Preisliste mit Gewerbeanmeldung an.

Wir suchen auch weiterhin noch Stützpunkthändler mit Gebietsschutz in ganz Europa. Interessenten aus dem Ausland wollen bitte die 14 % MwSt aus unseren Angebotspreisen wieder abziehen. Die Preise verstehen sich ab Werk und ausschließlich Versandverpackung.

ALBATROS-COMPUTER GmbH

7107 BAD WIMPFEN, MAGNUSSTR. 9, TEL. 0 70 63/70 57 (KEIN LADENVERKAUF)

➔ EIN NAME, DEN MAN SICH MERKEN SOLLTE. ➔

PC-COLLEGE

INSTITUT FÜR EDV-AUSBILDUNG
BERLIN · STUTTGART



Setzen Sie Ihren PC noch effektiver ein – wir zeigen Ihnen wie !

- Schulungen aller gängigen Softwareprodukte
- Firmenseminare (auch inhouse)
- Telefon-Hotline-Service
- Anwender-Programmierung
- Kaufm. Unternehmensberatung

Bitte Seminarprogramm anfordern !

Tauentzienstr. 5 - 1000 Berlin 30 030 / 24 98 96
Calwer Str. 17 - 7000 Stuttgart 1 0711 / 226 20 03

Immer am Drücker: Tastatur-Interrupt 16hex

Wenn Sie ganz genau wissen wollen, welche Tasten gleichzeitig betätigt wurden, beziehungsweise gedrückt sind, dann hilft Ihnen der Tastatur-Interrupt 16hex dabei, dies festzustellen.

Während die Funktionen des MS-DOS-Interrupt 21 nur ASCII-Zeichen einlesen können, bietet der Interrupt 16 einiges mehr. Über den Tastatur-Interrupt 16hex lässt sich beispielsweise feststellen, ob eine Taste zusammen mit [Shift], [Ctrl] oder [Alt] gedrückt wurde und welche Sonder-taste gegenwärtig aktiv ist. Tabelle 1 zeigt die drei Funktionen des Interrupt 16hex und Tabelle 2 den Aufbau des Tastaturstatusbyte. Den Tastaturcode und dessen Bedeutung listet Tabelle 3 auf, und Tabelle 4 zeigt schließlich den erweiterten Tastaturcode.

Über den Tastatur-Interrupt 16hex lässt sich die Tastatur sehr hardwarenah abfragen

Im Gegensatz zu den Ausführungen in einigen IBM-Begleitmaterialien sind die Funktionen 00 und 01 sowohl für normale ASCII-Zeichen als auch für Sonderzeichen, wie beispielsweise die von den Funktions-tasten erzeugten Zeichen, verwendbar. Die Funktion 02 wird benutzt, um auf Änderungen des Tastaturstatus reagieren zu können. Nur mit dieser Funktion sind Sie beispielsweise in der Lage zu unterscheiden, welche der beiden Shift-Tasten gedrückt wurden.

(Markus Zietlow/ma)

Das Tastaturstatusbyte	
Bit	Bedeutung
Bit 0=1	Rechte Shift-Taste gedrückt
Bit 1=1	Linke Shift-Taste gedrückt
Bit 2=1	Ctrl-Taste gedrückt
Bit 3=1	Alt-Taste gedrückt
Bit 4=1	[Scroll Lock] gedrückt
Bit 5=1	[Num Lock] gedrückt
Bit 6=1	[Caps Lock] gedrückt
Bit 7=1	[Ins] gedrückt

Tabelle 2. Die dritte Funktion liefert im AL-Register den Status der Tastatur

Die Funktionen des Tastatur-Interrupt 16hex			
AH-Register (hex)	Eingabe	Ausgabe	Bedeutung
00	—	AX=Tastaturcode AL=0: AH=erweiterter Tastaturcode AL<>0: AH=Scan-Code, AL=ASCII-Code	Lies Tastaturcode aus Puffer; warte auf Taste, wenn Puffer leer ist
01	—	Z-Flag=0: AX=Tastaturcode (Pufferzeiger unverändert) AL=0: AH=erweiterter Tastaturcode AL<>0: AH=Scan-Code; AL=ASCII-Code Z-Flag=1: Puffer leer	Prüfe Tastaturcode auf der Pufferzeiger-Position, und prüfe, ob Puffer leer ist
02	—	AL=Tastaturstatus	Lies Tastaturstatus (siehe Tabelle 2)
Der Tastatur-Interrupt 16hex beeinflusst nur AX und die Statusbits			

Tabelle 1. Der Tastatur-Interrupt 16hex stellt drei Funktionen zur Verfügung

Die Tastaturcodes des PC (Teil I) (Hex-Werte in Klammern)					
AL	Zeichen	[Shift] gedrückt	[Ctrl] gedrückt	[Alt] gedrückt	
01	Esc (1B)	Esc (1B)	Esc (1B)	Esc (1B)	
02	1	!	*	**	
03	2	"	*	**	
04	3	\$	*	**	
05	4	\$	*	**	
06	5	%	*	**	
07	6	&	*	**	
08	7	/	*	**	
09	8	(	*	**	
0A	9	)	*	**	
0B	0	=	*	**	
0C	B	?	*	**	
0D	,	,	*	**	
0E	BS (08)	BS (08)	DEL (7F)	*	
0F	HT (09)	HT back	*	*	
10	q	Q	DC1 (11)	**	
11	w	W	ETB (17)	**	
12	e	E	ENQ (05)	**	
13	r	R	DC2 (12)	**	
14	t	T	DC4 (14)	**	
15	z	Z	SUB (1A)	**	
16	u	U	NAK (15)	**	
17	i	I	HT (09)	**	
18	o	O	SI (0F)	**	
19	p	P	DLE (10)	**	
1A	ü	Ü	*	*	
1B	+	*	*	*	
1C	CR (0D)	CR (0D)	LF (0A)	*	
1D	[Ctrl]				
1E	a	A	SOH (01)	**	
1F	s	S	DC3 (13)	**	
20	d	D	EOT (04)	**	
21	f	F	ACK (06)	**	
22	g	G	BEL (07)	**	
23	h	H	BS (08)	**	
24	j	J	LF (0A)	**	
25	k	K	VT (0B)	**	
26	l	L	FF (0C)	**	
27	ö	Ö	*	*	
28	ä	Ä	*	*	
29	#	↑	*	*	
2A	[Shift]				

2B	<	>	*	*
2C	y	Y	EM (19)	**
2D	x	X	CAN (18)	**
2E	c	C	ETX (03)	**
2F	v	V	SYN (16)	**
30	b	B	STX (02)	**
31	n	N	SO (0E)	**
32	m	M	CR (0D)	**
33	,	;	*	*
34	.	:	*	*
35	—	—	US (1F)	*
36	[Shift]			
37	*	[Print Screen]	**	*
38	[Alt]			
39	SP (20)	SP (20)	SP (20)	SP (20)
3A	[Caps Lock]			
45	[Num Lock]			
46	[Scroll Lock]			

AL	Zeichen	[Num Lock] gedrückt	[Ctrl] gedrückt	[Alt] gedrückt
47	[Home]	7	Bildschirm löschen	**
48	[Cursor Up]	8	*	**
49	[Page Up]	9	Textanfang + [Home]**	
4A	.	4	Ein Wort zurück	*
4B	[Cursor Left]		*	**
4C	*	5	*	*
4D	[Cursor Right]	6	Ein Wort vorwärts	**
4E	+	+	*	*
4F	[End]	1	Bis Zeilenende löschen	**
50	[Cursor Down]	3	*	**
51	[Page Down]		Bis Bildschirmende löschen	**
52	[Insert]	0	*	*
53	[Delete]		*	*

*: keine Funktion
**: erweiterter Tastaturcode in AX

Tabelle 3. Der Tastaturcode zeigt an, welche Taste beziehungsweise welche Tastenkombination gedrückt wurde

Wissen Sie mehr von PCs als andere Leute?

Dann schreiben Sie uns. Die Redaktion DOS International sucht aktive Mitarbeiter, die über interessante Gebiete aus der »Computerei« berichten wollen. MS-DOS, DOS Plus, GEM, Drucker, Basteleien – es gibt nichts, worüber nicht andere Leser unserer Zeitschrift auch etwas lesen wollen.

Ihren Einsatz lassen wir uns natürlich etwas kosten. Ihr Manuskript wird selbstverständlich honoriert. Bilder, Zeichnungen, Tabellen und Listings werden gesondert berechnet.

Doch bevor Sie sich hinsetzen mit dem Schreiben beginnen: Rufen Sie uns an.

Sagen Sie uns, was Sie schreiben wollen, und wir klären, ob – und wann – der Artikel in unser Konzept paßt.

Der Artikel muß mit einem Textsystem erfaßt werden. Wir brauchen von Ihnen eine Diskette im MS-DOS-Format, auf der der Text als reine ASCII-Datei steht.

Aber auch wenn Sie nur Vorschläge für Themen haben, dann schreiben Sie uns.

DOS International ist nämlich eine Zeitschrift von aktiven PC-Besitzern für aktive PC-Benutzer.

Besonders das DOS-Forum lebt von Ihrer Mitarbeit. Auch die Redaktion kann nicht alle Antworten wissen. Fragen an uns, die nicht sofort beantwortet werden können, werden im DOS-Forum veröffentlicht. Wissen Sie eine Antwort, so wird diese veröffentlicht.

Sie sehen: DOS International ist die PC-Zeitschrift, in der das Wissen der PC-Benutzer untereinander ausgetauscht wird.

**Redaktionsbüro
Everts & Hagedorn
Redaktion DOS International
Hans-Stießberger-Str. 3
8013 Haar/München
Tel.: (089) 46 10 01-0**

BIOS

Die Tastaturcodes des PC (Teil II)	
AX	Funktion
0300	NUL (00)
0F00	TAB back
1000	[Alt-Q]
1100	[Alt-W]
1200	[Alt-E]
1300	[Alt-R]
1400	[Alt-T]
1500	[Alt-Z]
1600	[Alt-U]
1700	[Alt-I]
1800	[Alt-O]
1900	[Alt-P]
1E00	[Alt-A]
1F00	[Alt-S]
2000	[Alt-D]
2100	[Alt-F]
2200	[Alt-G]
2300	[Alt-H]
2400	[Alt-J]
2500	[Alt-K]
2600	[Alt-L]
2C00	[Alt-Y]
2D00	[Alt-X]
2E00	[Alt-C]
2F00	[Alt-V]
3000	[Alt-B]
3100	[Alt-N]
3200	[Alt-M]
3B00	[F1]
3C00	[F2]
3D00	[F3]
3E00	[F4]
3F00	[F5]
4000	[F6]
4100	[F7]
4200	[F8]
4300	[F9]
4400	[F10]
4700	[Home]
4800	[Cursor Up]
4900	[Page Up]
4B00	[Cursor Left]
4D00	[Cursor Right]
4F00	[End]
5000	[Cursor Down]
5100	[Page Down]
5200	[Insert]
5300	[Delete]

5400	[Shift-F1]
5500	[Shift-F2]
5600	[Shift-F3]
5700	[Shift-F4]
5800	[Shift-F5]
5900	[Shift-F6]
5A00	[Shift-F7]
5B00	[Shift-F8]
5C00	[Shift-F9]
5D00	[Shift-F10]
5E00	[Ctrl-F1]
5F00	[Ctrl-F2]
6000	[Ctrl-F3]
6100	[Ctrl-F4]
6200	[Ctrl-F5]
6300	[Ctrl-F6]
6400	[Ctrl-F7]
6500	[Ctrl-F8]
6600	[Ctrl-F9]
6700	[Ctrl-F10]
6800	[Alt-F1]
6900	[Alt-F2]
6A00	[Alt-F3]
6B00	[Alt-F4]
6C00	[Alt-F5]
6D00	[Alt-F6]
6E00	[Alt-F7]
6F00	[Alt-F8]
7000	[Alt-F9]
7100	[Alt-F10]
7200	[Ctrl-Print Screen] (Hardcopy ein/aus)
7300	[Ctrl-Cursor Left]
7400	[Ctrl-Cursor Right]
7500	[Ctrl-End]
7600	[Ctrl-Page Down]
7700	[Ctrl-Home]
7800	[Alt-1]
7900	[Alt-2]
7A00	[Alt-3]
7B00	[Alt-4]
7C00	[Alt-5]
7D00	[Alt-6]
7E00	[Alt-7]
7F00	[Alt-8]
8000	[Alt-9]
8100	[Alt-0]
8200	[Alt-`]
8300	[Alt-']
8400	[Ctrl-Page Up]

Tabelle 4. Der erweiterte Tastaturcode belegt das gesamte AX-Register

Der PC macht Druck: Drucker-Interrupt 17hex

Während die serielle Schnittstelle über den Interrupt 14hex angesteuert wird, ist für die parallele Schnittstelle der relativ einfach aufgebaute Drucker-Interrupt 17hex zuständig.

Wenn Sie in Assembler-Programmen Daten auf den Drucker ausgeben wollen, empfiehlt sich der Drucker-Interrupt 17hex (Tabelle 1). Dieser Interrupt stellt dem Programmierer drei Funktionen zur Verfügung. Eine Funktion initialisiert den Drucker, eine weitere sendet ein ASCII-Zeichen an die parallele Schnittstelle (an die der Drucker angeschlossen ist) und die dritte Funktion überprüft den Druckerstatus. Das Druckerstatusbyte (Tabelle 2) informiert den An-

Die Funktionen des Drucker-Interrupt 17hex			
AH-Register	Eingabe	Ausgabe	Bedeutung
0	AL = Zeichen DX = Drucker (0 bis 2)	AH = Druckerstatus	Sende ein Zeichen an den Drucker
01	DX = Drucker (0 bis 2)	AH = Druckerstatus	Initialisiere Drucker
02	DX = Drucker (0 bis 2)	AH = Druckerstatus	Lies Druckerstatus

Der Drucker-Interrupt 17hex beeinflusst nur das Register AH

Tabelle 1. Mit dem Drucker-Interrupt 17hex steuern Sie die parallele Schnittstelle an

wender darüber, ob das gesendete ASCII-Zeichen empfangen und erfolgreich verarbeitet wurde. Weil die Zeitspanne, in der das an die parallele Schnittstelle angeschlossene Gerät auf den Empfang eines Zeichens reagieren muß, genau festgelegt ist, führt ein Überschreiten dieser Wartezeit (Time Out) zu einem Funktionsabbruch mit einer Fehlermeldung in Bit 0 des Druckerstatusbyte. Der für diesen Fall übliche Standardfehler: Der Drucker wurde nicht in den On-line-Zustand versetzt.

(Markus Zietlow/ma)

Das Druckerstatusbyte	
Bit	Bedeutung
Bit 0 = 1	Drucker reagiert nicht, Wartezeit überschritten (Time Out)
Bit 1 = 0	(konstant)
Bit 2 = 0	(konstant)
Bit 3 = 1	Ein-/Ausgabefehler (Error)
Bit 4 = 0	Drucker nicht ausgewählt (Offline)
Bit 4 = 1	Drucker ausgewählt (Online)
Bit 5 = 1	Papierende erreicht (Paper Out)
Bit 6 = 1	Empfangsbestätigung (Acknowledge)
Bit 7 = 0	Drucker beschäftigt (Busy)
Bit 7 = 1	Drucker nicht beschäftigt

Tabelle 2. Alle Funktionen des Drucker-Interrupt geben auch ein Druckerstatusbyte zurück

Zeit im Griff: Systemuhr-Interrupt 1Ahex

Über den Systemuhr-Interrupt 1Ahex lassen sich der Zeitzähler des PC und die Echtzeituhr des AT direkt ansprechen. Auf diese Weise haben Sie mit dem Systemuhr-Interrupt immer die Zeit im Griff.

Während sich über die Funktionen 2C und 2Dhex des MS-DOS-Interrupt 21hex die Uhrzeit des Computers in Stunden, Minuten und Sekunden lesen und setzen läßt, bietet der Systemuhr-Interrupt 1Ahex beim PC deutlich weniger Komfort. Hier bestimmt lediglich ein 32-Bit-Wert die Zeit. Doch dafür erhält man über den Systemuhr-Interrupt direkten Zugriff auf den Zeitzähler des PC, der sich für präzise Zeitmessungen eignet.

Der Zeitzähler arbeitet im PC mit dem Takt des Ticker-Interrupt 18hex, das heißt, er erhöht den Zählerwert genau 18,2065mal in der Sekunde. Die Zählung der Impulse beginnt bei Mitternacht. Die Impulse werden durch den Software-Interrupt 08 erzeugt, der regelmäßig von einem Oszillator des Zeitgeber-Bausteins 8253 aufgerufen wird. Wenn ein Tag vergangen beziehungsweise ein 24-Stunden-Intervall abgelaufen ist, speichert der Computer dieses Ereignis im AL-Re-

Die Funktionen des Systemuhr-Interrupt 1Ahex				
AH-Register (hex)	Computer	Eingabe	Ausgabe	Bedeutung
00	PC/XT/AT	—	AL=0: kein 24-Stunden-Intervall vergangen AL < > 0: 24-Stunden-Intervall vergangen DX = untere Hälfte des 32-Bit-Zeit Zählers	Lies Zeitzähler (Zeitähler/18,2 = Sekunden) CX = obere Hälfte des 32-Bit-Zeit Zählers
01	PC/XT/AT	CX = obere Hälfte des 32-Bit-Zeit Zählers DX = untere Hälfte des 32-Bit-Zeit Zählers	—	Setze Zeitzähler (Zeitähler/18,2 = Sekunden)
02	AT	—	C-Flag=0: Batterie korrekt CH=Stunde CL=Minute DH=Sekunde (alle Zeitangaben im BCD-Format) C-Flag=1: Batterie der Uhr ist leer	Lies die Zeit aus der Echtzeituhr
03	AT	CH=Stunde CL=Minute DH=Sekunde DL=Jahreszeit DL=0: keine Sommerzeit DL=1: Sommerzeit (alle Zeitangaben müssen im BCD-Format vorliegen)	—	Setze die Zeit in der Echtzeituhr
04	AT	—	C-Flag=0: Batterie korrekt CH=Jahrhundert (19 oder 20) CL=Jahr DH=Monat DL=Tag (Zeitangaben im BCD-Format) C-Flag=1: Batterie der Uhr ist leer	Lies das Datum aus der Echtzeituhr
05	AT	CH=Jahrhundert (19 oder 20) CL=Jahr DH=Monat DL=Tag (alle Zeitangaben müssen im BCD-Format vorliegen)	—	Setze das Datum in der Echtzeituhr
06	AT	CH=Stunde CL=Minute DH=Sekunde (alle Zeitangaben müssen im BCD-Format vorliegen)	—	Setze die Alarmzeit in der Echtzeituhr
07	AT	—	—	Lösche die Alarmzeit

Tabelle. Die Funktionen des Interrupt 1Ahex kontrollieren die Zeitmessung im Computer

gister ab. Es ist allerdings nicht möglich, über das AL-Register die Anzahl der vergangenen 24-Stunden-Intervalle festzustellen. Auf einem AT lassen sich über den Systemuhr-Inter-

rupt 1Ahex zusätzlich die Uhrzeit, das Datum und die Alarmzeit der eingebauten Echtzeituhr lesen und setzen.

(Markus Zietlow/ma)

Den Bildschirmkarten aufs Register geschaut

Grafik gehört zu den faszinierenden Anwendungsbereichen der Computertechnik. Doch gerade beim PC ist die Anzahl der verschiedenen Grafikkarten sehr groß. Im folgenden werden die Unterschiede zwischen den Standards MDA, Hercules, CGA, EGA und VGA sowie deren Programmierung erläutert.

Grafik spielt in der Computertechnik eine gewichtige Rolle, da optische Eindrücke den Menschen stark beeinflussen. Bei interessanter Aufbereitung zieht Computergrafik viel Aufmerksamkeit auf sich und übt große Faszination auf den Betrachter aus. Auf kommerziellem Gebiet eignet sich Grafik hervorragend zur leicht verständlichen Darstellung von Zahlen, im Heimbereich ermöglicht sie hingegen faszinierende Computerspiele.

Die Ansprüche an Auflösung und Anzahl der darstellbaren Farben hat mit der Zeit zugenommen. Für den PC gibt es heute fünf Grafikkarten, die sich als Standard etabliert haben. Dieser und die vier folgenden Artikel erläutern Ihnen die Unterschiede und führen Sie in die Programmierung der Bildschirm-Adapter ein.

Bei der Ausgabe auf dem Bildschirm sind mehrere Hard- und Software-Komponenten beteiligt, die genau aufeinander abgestimmt sein müssen, um eine störungsfreie Darstellung zu garantieren. Für die Bilderzeugung sind drei wichtige Teile des PC von Bedeutung: der Monitor, die Steuerlogik und der sogenannte Bildschirmspeicher. Dazu kommt noch die Software beziehungsweise das Betriebssystem, dessen Routinen die Bildschirmausgabe maßgeblich beeinflussen. Die Beziehung zwischen den einzelnen Komponenten ist in Bild 1 aufgezeichnet.

Die Steuerlogik ist der zentrale Baustein der Grafikdarstellung

Die Steuerlogik benutzt die Daten im Bildschirmspeicher, um das Signal für den Monitor zu erzeugen. Zuletzt wandelt der Monitor den vom Signal-generator erzeugten elektrischen Impuls in Licht um.

Normalerweise sind Bildschirmsteuerlogik und Bildschirmspeicher zusammen auf dem Bildschirm-Adapter, auch Grafikkarte genannt, angebracht. Die Steuerlogik, deren Aufbau Sie Bild 2 entnehmen, holt die

digitalen Daten aus dem Bildschirmspeicher, übersetzt die Zeichen in Punktmuster und wandelt die digitalen Punkte in ein Signal für den Monitor um.

Der zentrale Baustein, den man als Video-Controller bezeichnet, bestimmt die Fähigkeiten der Steuerlogik. Er liest die Daten aus dem Bildschirmspeicher und kontrolliert die Funktionen der übrigen Komponenten. Der Zeichengenerator wandelt ein Zeichen vom ASCII-Format in das Punktmuster um, das Sie dann zu sehen bekommen. Abschließend gibt das Schieberegister die Punkte in der korrekten Reihenfolge an den Signal-Generator weiter.

Der Signal-Generator wandelt die Informationen vom Schieberegister in ein Monitorsignal um. Die Verarbeitung der Bildschirminformationen erfolgt mit einer sehr hohen Geschwindigkeit. Ein Grafikbildschirm mit einer Auflösung von 640x200 Punkten enthält insgesamt 128000 Bildpunkte. Da das Bild mindestens 50mal in der Sekunde ausgegeben wird, ergibt das 6,4 Millionen Punkte pro Sekunde. In dieser Rechnung ist die Zeit nicht berücksichtigt, während der der Bildschirm zur Neuausrichtung des Elektronenstrahls kurz abgeschaltet wird. So überrascht es auch nicht, daß die Übertragungsgeschwindigkeit von der Steuerlogik zum Bildschirm bei Berücksichtigung dieser Ausfallzeiten über 10 Millionen Punkte pro Sekunde beträgt.

Die erste Komponente der Bilderzeugung ist der Bildschirmspeicher, der – wie Sie bereits wissen – die digitalen Daten aufnimmt. Dieser Speicherbereich wird wie der normale Arbeitsspeicher adressiert; das bedeutet, daß man ohne Schwierigkeiten Daten in ihm ablegen und aus ihm lesen kann.

Der Unterschied zwischen Text- und Grafikdarstellung liegt allein in der Form der Abbildung. Text wird im Bildschirmspeicher als Folge von ASCII-Zeichen abgelegt. Jedem Zeichen folgt zusätzlich ein sogenanntes Attribut-Byte, das die Farbe oder die Darstellungsform bestimmt. Bei der

Grafikausgabe werden im Bildschirmspeicher die Daten der einzelnen Punkte abgelegt. Wie bei der Textdarstellung bestimmt die Lage eines Punktes im Bildschirmspeicher dessen Position auf dem Bildschirm. In dem etwas vereinfacht dargestellten Verfahren ist, abhängig von der Anzahl der Farben, ein Punkt in einem, zwei oder mehreren Bytes abgespeichert.

Neben der Hardware spielt die Software eine wichtige Rolle bei der Bildschirmdarstellung. Es gibt vier Methoden, Daten auf den Monitor zu bringen. Der einfachste Weg, ein Zeichen auszugeben, ist die Verwendung eines Kommandos in einer höheren Programmiersprache (beispielsweise »print« in GW-Basic, »write« in Pascal oder »printf« in C). Die Ausgabe auf dieser Ebene bietet einige Vorteile. Erstens bleibt die Ausgabe – soweit die Programmiersprache auf andere Computer umgesetzt wurde – portabel, das Programm funktioniert also auch auf anderen Systemen. Zweitens ist eine Funktion einer höheren Programmiersprache leicht verständlich. Das erleichtert sowohl die Konzeption als

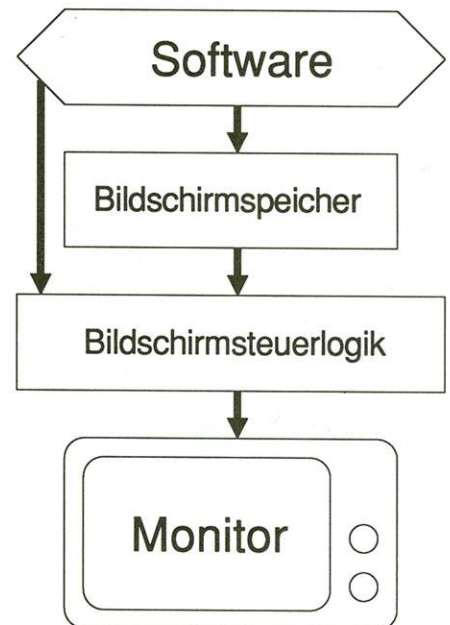


Bild 1. Monitor und Computer bilden bei der Monitorausgabe eine kompakte Einheit

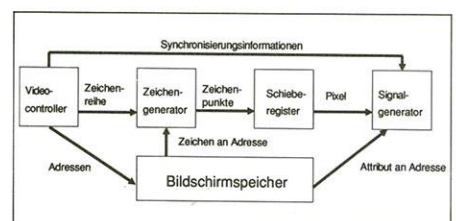


Bild 2. Die Steuerlogik kontrolliert den Bildschirmaufbau

MS-DOS Anwendungen für Profis – und solche, die es werden wollen

BCi DisAsm erschließt jedes Programm!

BCi DisAsm – Der intelligente Disassembler

BCi DisAsm ist ein intelligenter, selbstständig dokumentierender MS-DOS Disassembler. Die ausgefeilten Routinen im BCi DisAsm analysieren in kürzester Zeit selbst komplexen und umfangreichen Programmcode und zeigen dabei potentielle Problempunkte auf.

BCi DisAsm – Das Lernmittel

Sie brauchen kein Spezialist in Assembler-Programmierung zu sein. BCi DisAsm produziert aus jedem beliebigen Programm ihres IBM-kompatiblen Computers ein gut dokumentiertes, leicht verständliches Assemblerlisting.

BCi DisAsm – Einfach in der Handhabung

BCi DisAsm kann sowohl über Parameter von der DOS-Kommandozeile aus als auch interaktiv über Menüs gesteuert werden. Es besteht keine Notwendigkeit, neue Befehle zu lernen oder ständig im Handbuch nachzuschlagen.

Minimale Systemanforderungen

256 kB RAM, 8088/8086/80186/80286 oder 80386 CPU, MS-DOS oder PC-DOS Version 2.0 oder größer

BCi DisAsm wird Ihnen bei den folgenden Schritten helfen:

1. Lernen Sie Assembler-Programmierung, wenn Sie möchten.
2. Entdecken Sie, warum Programm XYZ auf ihrem Rechner nicht läuft und tun Sie es was dagegen.
3. Ändern oder entfernen Sie Programmteile.
4. Sichern Sie sich gegen destruktive Programme ab. Finden Sie einen "Virus" und entfernen Sie ihn.
5. Integrieren Sie Code aus kompilierten Programmen in eigene Anwendungen.
6. Erhöhen Sie die Kompatibilität Ihrer Software.
7. Passen Sie Software an verschiedene MS-DOS-Versionen an.

© 1987 BCi, West Germany und USA · Lotus ist eingetragenes Warenzeichen der Lotus Corp. MS-DOS ist eingetragenes Warenzeichen der Microsoft Corp.

Best.-Nr. 227 5 1/4" Disk.

Best.-Nr. 233 3 1/2" Disk.



Intelligenter Disassembler Version 1.0

199, – DM*

(unverbindliche Preisempfehlung)

CGX v 1.2

Turbo Pascal Toolbox der Superlative

CGX (Color Grafik EX tension) ist eine Sammlung von über 100 leistungsfähigen Routinen, mit denen grafisch beeindruckende und benutzerfreundliche Turbo Pascal-Programme sehr einfach geschrieben werden können.

Mit CGX können aus ein und demselben Quelltext lauffähige Programme für 3 verschiedene hochauflösende Grafikmodi erzeugt werden.

EGA Farbgrafik mit 16 Farben · PC 1512 Spezialmodus mit 16 Farben · CGA Hires-Grafik mit 2 Farben

Die CGX-Prozeduren sind durch Verwendung modernster Algorithmen mit Assembler äußerst effektiv.

Die **KERNEL**-Bibliothek stellt den CGX-Betriebssystemkern mit einem neuen Bildschirmtreiber dar. Dies ermöglicht verschiedene Schrifttypen mit Unterstreichen, Fettschrift, Exponenten und Indizes und vieles mehr.

Die **GRAPHIK**-Bibliothek enthält zahlreiche Grafik-Prozeduren mit Features wie Ellipsenbögen, Füllprozeduren mit Mustern und über 100 Farben, Verzerren und Biegen von Bildschirmausschnitten.

Die **MOUSE**-Bibliothek erlaubt einfache Erstellung von mausgesteuerten Programmen.

Mit der **HARDCOPY**-Bibliothek können Ausdrucke beliebiger Bildausschnitte in frei wählbaren Formaten erstellt werden.

Die **SCREEN**-Bibliothek erlaubt das Ablegen von Bildschirmbereichen im Hauptspeicher oder auf Diskette.

Mit der **WINDOW**-Bibliothek steht ein System zur Verwaltung von Text- und Grafikfenstern zur Verfügung, welche die Grundlagen einer modernen Menutechnik darstellen.

Neu: Die Bibliothek **CXKEYSIM** ermöglicht, mittels Cursortasten eine Maus zu simulieren.

Im Lieferumfang enthalten:

Umfangreiches deutsches Handbuch · Füllmustereditor im Quellcode · Zeichensatzeditor im Quellcode · Dokumentierter Quellcode des Fraktalgenerators

Konfiguration: Turbo Pascal 3.0 auf MS-DOS, unterstützte Grafikmodi: EGA-, CGA- oder PC 1512 Grafik-karte

Best.-Nr. 224 5 1/4" Disk.

Best.-Nr. 223 3 1/2" Disk.

149, – DM*

(unverbindliche Preisempfehlung)

Update-Service:

Bei Zusendung Ihrer CGX-Originaldiskette erhalten Sie das Update von Version 1.0 auf 1.2 zum Preis von:

Best.-Nr. 244 5 1/4" Disk.

Best.-Nr. 245 3 1/2" Disk.

24, – DM*

Die BASIC2-Toolbox

Das Buch zu BASIC2

PC 1512/1640-Besitzer können aufatmen. Jetzt ist sie da, die BASIC2-Toolbox. Mit diesem einzigartigen Buch sind Sie nun in der Lage, das Locomotive BASIC2 besser auszunutzen, die Programmierung wird zum Kinderspiel.

Die BASIC2-Toolbox bietet nicht die x-te Einführung, sondern gibt anhand leistungsstarker und praxisorientierter Routinen und Programme den nötigen Durchblick. Schritt für Schritt lernen Sie die strukturierte Programmentwicklung bis hin zum fertigen und lauffähigen Programm. Der Text ist in leicht und für jedermann verständlicher Form geschrieben, der Lerneffekt ist quasi garantiert. Alle Beispielprogramme sind sofort nachvollziehbar und stammen aus der täglichen Anwendungspraxis.

Einige Beispiele aus dem Inhalt:

- Kurze Einführung in die Software-Entwicklung
- Werkzeuge für BASIC2
- Werkzeuge zur Behandlung von Text-dateien
- Werkzeuge zur Software-Entwicklung
- Ein kompletter Disassembler für den Intel 8086



Die BASIC2-Toolbox
Autor: Günter Born,
250 Seiten, 54 Abb.,
ISBN Nr. 3-926177-01-2

(Best.-Nr. 402)

nur **49, – DM***

FontEdit Pc

Universeller Zeichensatz- und Symboleditor

Eigene Zeichensätze selbst erstellt

Vektorzeichensätze, 100% kompatibel zum Borlands Graphics Interface (BGI) · bis zu 256 Zeichen in einem Zeichensatz (extended) · Zeichen beliebig vergrößerbar und drehbar · voll menuegesteuert · Zeicheneditor mit horizontalem und vertikalem Spiegeln · nun auch Umlaute und beliebige Sonderzeichen möglich.

Im Lieferumfang

Drei verschiedene Zeichensätze mit Umlauten und '§' · Griechischer und gotischer Zeichensatz · Zeichensätze für sehr große und sehr kleine Schrift · Zeichensatz "Script" · Kursiver "Sanserif"-Zeichensatz · Beispielprogramm zur Anwendung grafischer Objekte · Ausführliches Handbuch mit Beschreibung des Aufbaus der Vektorzeichensätze, deren Header und Eckwerte · Beschreibung der Routinen.

Symbole für Grafikprogramme & Spiele

Spelfiguren als Vektor-Objekte · Grafiken mittels BGI gestaltet · Auch Animationen (Arcadespiele) mittels "GetImage" und "PutImage" möglich.

Konfiguration:

PC XT/AT mit 384 kByte oder mehr, 1 Diskettenlaufwerk bzw. Festplatte, MS-DOS ab Version 2.0, CGA-, MCGA-, EGA-, VGA-, AT&T mit 400 Zeilen, 3270 PC und Herculeskarten werden unterstützt. Nutzbar mit Turbo Pascal 4.0, Turbo C 1.5, Turbo Prolog 2.0. Mittels der mitgelieferten Treiber Routinen auch anwendbar mit anderen grafikfähigen Pascal- und C-Compilern.

Best.-Nr. 242 5 1/4" Disk.

Best.-Nr. 243 3 1/2" Disk.

99, – DM*

(unverbindliche Preisempfehlung)



bringt Ihrer Tastatur das Zaubern bei

Legen Sie ganze Texte auf 1 Taste!
Aktivieren Sie Befehlsfolgen mit 1 Tastendruck!

Recorderfunktion zur Makroaufzeichnung · mit "Copy Cursor" wird Bildschirmtext zum Makro · gezieltes Abspeichern mit Kommentaren · komfortabelste Editiermöglichkeiten aller Makros · verschiedene Makros auf 1 Taste · "Chaining": Makros rufen sich gegenseitig auf · "Sendkeys": Makros in der Batch-Datei · Hyperkey läuft im Hintergrund anderer Programme · mit Makros für Basic- und Pascalprogrammierer

mit TSR-Verwaltung

Auflisten aller geladenen speicherresidenten Programme · Angabe ihres Speicherplatzbedarfs und der verwendeten Interruptvektoren · Berechnung des noch zur Verfügung stehenden Speicherplatzes · Markieren von residenten Programmen beim Ladevorgang · Gefahrloses Entfernen
Alle Programme mit umfangreicher Dokumentation.

Best.-Nr. 229 5 1/4" Disk.

Best.-Nr. 235 3 1/2" Disk.

99, – DM*

(unverbindliche Preisempfehlung)

PASCAL SONDERDRUCK

Das Beste aus PASCAL 11/86 bis 6/87

Aus dem Inhalt:

Kurvendiskussion · Mathematische Funktionen selbst erweitert · Rechnen mit komplexen Zahlen in Turbo Pascal · Gleichungssysteme und Ausgleichspolynome · Strings unter Wirth'schem Standard selbstgemacht · Press & Unpress, Dateien komprimiert abspeichern · GSX – Die unbekannte Größe bei CP/M 3.0 · Source-Code Referenz Lister · Dreidimensionale Darstellungen in Pascal ST+ (Atari ST) · Bibliothek zum Rechnen mit Matrizen in Turbo Pascal · Simulation in Pascal, ein Fahrkartenautomat · Ähnlichkeitsbestimmung von Zeichenketten · Pascal Colleg · Vom Punkt zur dritten Dimension · Zweidimensionale Meßwertdarstellung in Pascal · Plotter-Simulator für den Matrixdrucker · Externe Kommandos in Turbo Pascal 3.0 · Menügesteuerte Druckereinstellung · Datenfernübertragungsprogramm in Pascal · HGR-Copy: Bildschirmausschnitte und Zoom bei Hardcopies · Interrupts · Der Case-Trick, PEEK und POKE für Pascal-Programmierer

Alle Programme sind in der Praxis getestet, Nachträge und Verbesserungen wurden eingearbeitet. Sie sind, mit einer Ausnahme, unter Turbo Pascal 3.0 lauffähig und eignen sich für alle Hardware und Betriebssysteme vom Homecomputer bis zum Mini.

PASCAL Sonderdruck Best.-Nr. 345

25, – DM

Alle Programme sind auf Datenträger in der abgedruckten Version erhältlich:

PASCAL Sonderdruck Volume 1 und 2

MS-DOS 5 1/4" Best.-Nr. 346 **35, – DM** Atari ST 3 1/2" Best.-Nr. 348 **35, – DM**

MS-DOS 3 1/2" Best.-Nr. 347 **35, – DM**

* Unabhängig von der Anzahl der bestellten Programme berechnen wir für das Inland 3, – DM bzw. für das Ausland 5, – DM Porto und Verpackung. Bitte benutzen Sie die Bestellkarte.

DMV · Abt. Software · Postfach 250 · 3440 Eschwege

auch die nachfolgende Verbesserung eines Programms erheblich. Die Ausgabe von Daten über eine höhere Programmiersprache weist aber zwei schwerwiegende Nachteile auf. Der größte Nachteil ist der Geschwindigkeitsverlust, der aufgrund der zahlreichen Überprüfungs- und Umrechnungsroutinen auftritt. Der zweite Nachteil besteht in der umständlichen und unvollständigen Unterstützung der Grafikfähigkeiten des PC, die die meisten höheren Programmiersprachen anbieten. Funktionen, die auf Maschinensprachenebene relativ leicht zu realisieren sind, erweisen sich in einer höheren Programmiersprache oft als schwierig oder sogar unmöglich.

Die Zeichenausgabe über die Routinen von MS-DOS ist sehr langsam und wenig flexibel

Eine andere Art der Zeichenausgabe bietet MS-DOS. Die MS-DOS-Ausgabe-Funktionen sind leider sehr einfach und langsam. Man kann nur Zeichen ausgeben, auf Farben oder irgendwelche anderen Attribute muß man verzichten. Diese Funktionen sind nicht einmal in der Lage, den Cursor an eine gewünschte Stelle zu positionieren. Der Vorteil der Funktionen liegt in ihrer Hardware- und Programmiersprachen-Unabhängigkeit.

Weitaus mehr Varianten und Funktionen bietet das BIOS (»Basic Input/Output System«) des Betriebssystems. Mit ihm können Sie den Bildschirm-Modus festlegen und abfragen, den Cursor positionieren, Texte mit dem zugehörigen Attribut ausgeben und einige Hilfsfunktionen ausführen. Zusätzlich gibt es Funktionen zum Setzen und Abfragen eines einzelnen Bildschirmpunktes, denn das BIOS ist auf einer niedrigen Ebene grafikfähig. Die Funktionen sind schnell und stellen alle notwendigen Fähigkeiten zur Verfügung. Aber auch die Unterprogramme des BIOS haben zwei Nachteile. Erstens sind die Aufrufe von der Syntax her umständlich und zu schwierig, um sie von einer höheren Programmiersprache aus direkt zu benutzen. Zweitens fehlen Funktionen, die komplexe Aufgaben wie das Zeichnen von Linien oder die Ausgabe eines Kreises erfüllen. Diese Funktionen muß man also selbst programmieren.

Die letzte Methode zur Bildschirmsteuerung ist bei weitem die schnellste. Wie eben erwähnt, kann man direkt in den Bildschirmspeicher schreiben. Dieser Weg ist aber auch der mit Abstand komplizierteste, denn Sie müssen jede Adresse selbst berechnen und das Bit-Muster für jedes Zeichen und jeden Punkt bestimmen. Außerdem kann es Probleme geben, wenn Sie Grafikkarten oder

Computer verwenden, die nicht 100prozentig kompatibel sind. Diese Methode ist also nicht hardware-unabhängig, sondern im Gegenteil für jede Grafikkarte verschieden. Der große Vorteil dieser Methode liegt in dem Geschwindigkeitsgewinn und in der Vielfältigkeit der Bildschirmausgabe.

Detaillierte Beschreibungen zu den einzelnen Grafikkarten finden Sie in den nachfolgenden Artikeln. Da sich

Das Attribut der Monochrom-Grafikkarte	
Bit-Kombination	Funktion
00000000	Unsichtbar
00000001	Unterstrichen
00000111	Normal
00001001	Leuchtend und unterstrichen
00001111	Leuchtend
01110000	Invers
10000001	Blinkend und unterstrichen
10000111	Blinkend
10001001	Blinkend, leuchtend und unterstrichen
10001111	Blinkend und leuchtend
11110000	Blinkend und invers

Tabelle 1. Das Monochrom-Attribut bestimmt die Form der Zeichendarstellung

Nummer	Modus	Grafikauflösung	Textauflösung	Farben	Adresse	Bildschirmkarte	Anmerkung
00	Text	320x200	40x25	16	B800	VGA, EGA, CGA	Auf CGA monochrom
00	Text	320x350	40x25	16	B800	VGA, EGA	
00	Text	360x400	40x25	16	B800	VGA	
01	Text	320x200	40x25	16	B800	VGA, EGA, CGA	Auf CGA monochrom
01	Text	320x350	40x25	16	B800	VGA, EGA	
01	Text	360x400	40x25	16	B800	VGA	
02	Text	640x200	40x25	16	B800	VGA, EGA, CGA	Auf CGA monochrom
02	Text	640x350	40x25	16	B800	VGA, EGA	
02	Text	720x400	40x25	16	B800	VGA	
03	Text	640x200	40x25	16	B800	VGA, EGA, CGA	Auf CGA monochrom
03	Text	640x350	40x25	16	B800	VGA, EGA	
03	Text	720x400	40x25	16	B800	VGA	
04	Grafik	320x200	-	4	B800	VGA, EGA, CGA	Auf CGA monochrom
05	Grafik	320x200	-	4	B800	VGA, EGA, CGA	
06	Grafik	640x200	-	2	B800	VGA, EGA, CGA	
07	Text	720x350	80x25	2	B000	VGA, EGA, MDA	Auch Hercules
07	Text	720x400	80x25	2	B000	VGA	
08							
09							nur auf IBM-PC-Junior
0A							nur auf IBM-PC-Junior
0B							nur auf IBM-PC-Junior
0C							reserviert für EGA-BIOS
0D	Grafik	320x200	-	16	A000	VGA, EGA	Für EGA-Karten mit 64 KByte RAM
0E	Grafik	640x200	-	16	A000	VGA, EGA	
0F	Grafik	640x350	-	2	A000	VGA, EGA	
10	Grafik	640x350	-	4	A000	EGA	Für EGA-Karten mit 64 KByte RAM
10	Grafik	640x350	-	16	A000	VGA, EGA	
11	Grafik	640x480	-	2	A000	VGA	
12	Grafik	640x480	-	16	A000	VGA	
13	Grafik	320x200	-	256	A000	VGA	

Tabelle 2. Die Videomodi der Bildschirmkarten garantieren Kompatibilität, da sie sich überschneiden

die Register und Adressen von Adapter zu Adapter geringfügig unterscheiden, sind die eben ausgeführten Erläuterungen nicht ohne Einschränkungen auf jede Bildschirmkarte anzuwenden.

Die MDA-Bildschirmkarte leistete Pionierarbeit, obwohl sie keinerlei Punktgrafik darstellen kann

Die erste Bildschirmkarte, die für den PC erhältlich war, ist der sogenannte Monochromadapter, den man mit der Bezeichnung MDA abkürzt. Der Monochromadapter wurde 1981 zusammen mit den ersten IBM PCs vorgestellt. Die Leistungsfähigkeit der Bildschirmkarte erweist sich als sehr eingeschränkt, da sie nicht grafikfähig ist. Der MDA kann ausschließlich die standardisierten ASCII-Zeichen und die Sonderzeichen aus dem IBM-Zeichensatz darstellen.

Diesen Nachteil gleicht der Monochromadapter allerdings mit einer hohen Qualität der Textdarstellung aus, die auch für moderne Grafikkarten Standard geblieben ist. Im Bildschirmspeicher des Monochromadap-

ters sind zwei Bytes für jedes Zeichen zuständig. Das erste Byte enthält den ASCII-Wert des jeweiligen Zeichens und das zweite das Attribut. Die Attributangabe, die in Tabelle 1 zusammengefaßt ist, bestimmt, ob ein Zeichen blinkend, invers, leuchtend, normal oder unterstrichen dargestellt wird. Der Bildschirmspeicher des Monochromadapters, der an der Speicherstelle B0000hex beginnt, ist 4 KByte groß. Das Betriebssystem nutzt aber nur 4000 (80x25x2) Byte des Speicherbereichs aus. Eine Bildschirmzeile von 80 Zeichen Länge belegt also aufgrund des zusätzlichen Attribut-Byte 160 Byte Speicherplatz. Die Adresse von Zeichen und Attribut können Sie mit den folgenden Formeln ermitteln:

Zeichen = &HB0000 + 160 x (zeile-1) + 2 x (spalte-1)

Attribut = &HB0001 + 160 x (zeile-1) + 2 x (spalte-1)

Der Monochromadapter hat heutzutage an Bedeutung verloren.

IBM hat sinnvollerweise seine verschiedenen Bildschirmadapter so konstruiert, daß sie immer kompatibel zu ihrem Vorgänger sind. Das heißt, die Programme, die für den Monochromadapter konzipiert wur-

den, laufen ohne Probleme sogar auf einer VGA-Karte, wenn die Bildschirmausgabe über das BIOS erfolgt. Die verschiedenen Bildschirmmodi und die zugehörigen Bildschirmkarten sind in Tabelle 2 zusammengestellt.

Wie Sie sicherlich bemerken, fehlt in der Aufstellung die Hercules-Grafikkarte. Das hat einen wichtigen Grund: Der Text-Modus der Hercules-Grafikkarte ist voll kompatibel zum Monochromadapter und wird aus diesem Grund auch mit Modus 7 angesprochen. Da die Hercules-Grafikkarte aber keine IBM-Entwicklung ist, fehlt der Grafik-Modus. Auf dieses Problem wird aber im Artikel über die Hercules-Grafikkarte näher eingegangen.

(Markus Zietlow/sw)

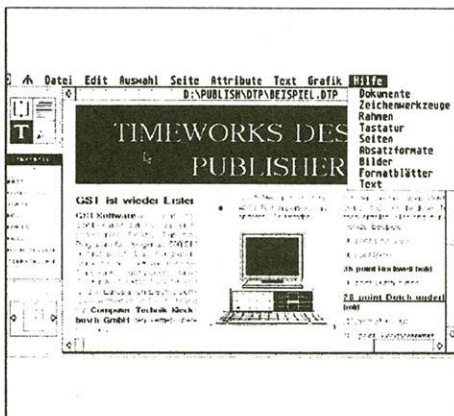
Literatur

- [1] IBM Personal System/2, Model 50 and 60 Technical Reference
- [2] Grundlagen von Grafik und Text beim PC, DOS Extra, Ausgabe 1
- [3] Alles über Grafikkarten, DOS Extra, Ausgabe 4
- [4] Video Systems, Microsoft Press
- [5] VGA – der neue Grafikstandard, DOS 1'88
- [6] EGA oder VGA?, DOS 3'88

TIMEWORKS

Einfache Handhabung
Seitenlayout
Text- und Grafikimport
Echtes DTP

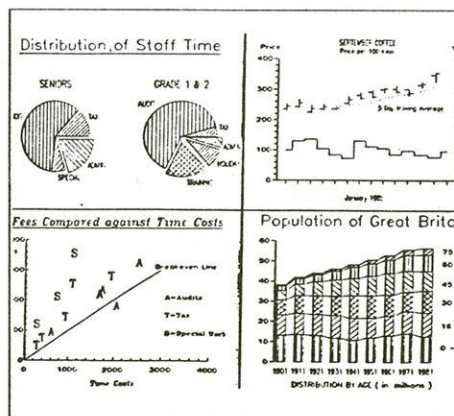
498,00 DM



LOGiSTiX

Tabellenkalkulation
Datenbank
Zeitplan/Netzplan
Grafiken

399,00 DM



DIVERSE

MS-DOS	DM
Flight Sim. III	129,00
Open Access 2.1	1999,00
Handy Scanner	598,00
A4-Scanner ab	2250,00
Chess	69,80
Star-Writer	398,00
Star-Planer	298,00
BS-Handel	1798,00
BS-Fibu	1798,00
BS-Timeadress	499,00
Wordstar	299,00
MS-Word 2.5	299,00
Multiplan 2.5	248,00

**Computer Technik
Kieckbusch GmbH**
Baumstammhaus
5419 Vielbach
☎ :02626-78336
FAX :02626-78337

Hercules

Der Held der Grafikwelt

Dem ersten IBM PC fehlte eine Bildschirmkarte, die Grafik mit einer hohen Auflösung darstellen konnte. IBM vertrat den Standpunkt, daß diese Anwendung für kommerzielle Programme unerheblich sei. Daß dem nicht so war, bewies die Firma Hercules, die als erste die Marktlücke entdeckte und mit der Hercules-Karte Pionierarbeit leistete. Der Erfolg der Karte ist daher nicht verwunderlich: Die Hercules-Karte wurde zur meistverkauften Grafikkarte für PCs, und ihre Inkompatibilität zu IBM-Grafikkarten war aufgrund großer Unterstützung durch die Softwarehäuser bald kein Hindernis mehr.

Als Weiterentwicklung des Monochromadapters konstruierte die Firma Hercules die inzwischen auf vielen Computern eingesetzte Hercules-Grafikkarte. Die Techniker von Hercules lösten das Problem, Text und Grafik auf dem gleichen Monitor abzubilden, indem Sie die Grundeigenschaften des Monochromadapters um Grafikfunktionen erweiterten. Die monochrome HGC-Karte (»Hercules Graphics Card«), die 1982 vorgestellt wurde, kann sowohl Grafik als auch alphanumerische Zeichen auf dem Monitor darstellen, der von IBM zusammen mit dem Monochromadapter ausgeliefert wurde.

Die Fähigkeit, eine Kombination aus Text und Grafik auf dem Bildschirm auszugeben, ist für viele Anwendungen von großer Bedeutung. Da viele PC-Anwender von diesem Leistungsmerkmal begeistert waren, überraschte der Erfolg der HGC-Karte keineswegs. Außerdem unterstützten viele große Software-Häuser die Bildschirmkarte, so daß sich der Hercules-Standard schon bald auf dem Markt etablierte.

Die Hercules-Karte enthält einen Schaltungsteil, der mit dem Monochromadapter nahezu identisch ist, und aus diesem Grund diese erste erhältliche Bildschirmkarte vollständig ersetzt. Im Gegensatz zu den 4000 Byte des Monochromadapters besitzt

die Hercules-Karte allerdings 64 KByte Bildschirmspeicher.

Indem die grundlegende Technik des Adapters verbessert und ergänzt wurde, hat man erreicht, daß dieser den Textmodus des Monochromadapters beherrscht und zusätzlich in einen Grafikmodus geschaltet werden kann, der eine Auflösung von 720x348 einfarbigen Bildpunkten erlaubt (Bild 1). Die Leistungsfähigkeit der Bildschirmkarte ist – bis auf die fehlende Farbdarstellung – befriedigend, da die Karte grafikfähig ist.

Die fehlenden Farben gleicht die Hercules-Grafikkarte mit einer hohen Auflösung von Text und Grafik aus

Die Hercules-Grafikkarte kann neben den standardisierten ASCII-Zeichen die Sonderzeichen aus dem IBM-Zeichensatz darstellen. Da die Zeichen der MDA- und der Hercules-Grafikkarte aus dem ROM gelesen werden, fehlt die freie Wahl eines Zeichensatzes. Das Ziel, eine Karte mit RAM-orientierten Zeichensätzen, wurde erst mit dem Nachfolger der Hercules-Grafikkarte, der im Juni 1986 vorgestellten HGC Plus (»Hercules Graphics Card Plus«), erreicht.

Die hohe Qualität der Textdarstellung bei der Hercules-Grafikkarte

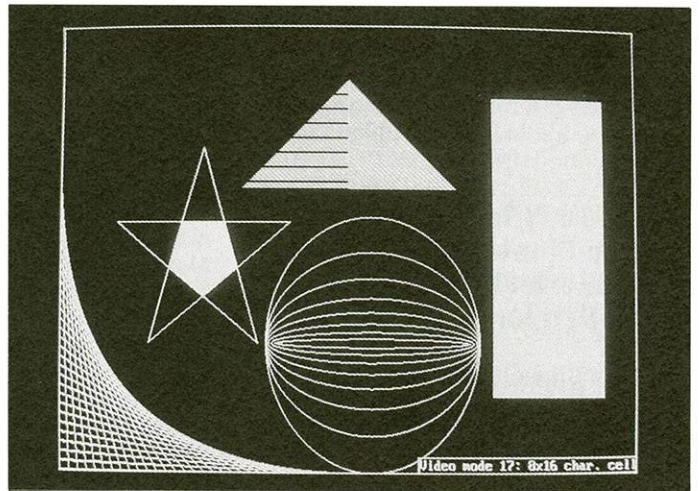


Bild 1. Für eine höhere Auflösung mußte man bei der Hercules-Grafikkarte auf Farbe verzichten

wird auch bei modernen Grafikkarten berücksichtigt. Eine Matrix aus 9x14 Bildpunkten bestimmt den Aufbau der ausgegebenen Zeichen. Auf eine Bildschirmseite passen 80 Zeichen in 25 Zeilen.

Im Bildschirmspeicher der Hercules-Grafikkarte sind – wie bereits beim Monochromadapter erwähnt – zwei Byte für jede Bildschirmposition zuständig. Das erste Byte enthält den ASCII-Wert des jeweiligen Zeichens und das zweite das Attribut. Die Attributangabe, die in Tabelle 1 des Artikels »Den Bildschirmkarten aufs Register geschaut« zusammengefaßt ist, bestimmt, ob ein Zeichen blinkend, invers, leuchtend, normal oder unterstrichen dargestellt wird. Der Bildschirmspeicher der Hercules-Grafikkarte, der an der Speicherstelle B0000hex beginnt, wird sowohl für Grafik als auch für die Textdarstellung verwendet.

Eine Bildschirmzeile von 80 Zeichen Länge belegt also aufgrund des zusätzlichen Attribut-Bytes 160 Byte Speicherplatz. Die Adresse von Zeichen und Attribut können Sie mit den folgenden Formeln ermitteln:

Zeichen = &HB0000 + 160 x (zeile-1) + 2 x (spalte-1)

Attribut = &HB0001 + 160 x (zeile-1) + 2 x (spalte-1)

Ein Nachteil der Hercules-Grafikkarte besteht darin, daß die Grafikdarstellung nicht vom BIOS unterstützt wird. Der Programmierer muß auf eigene Routinen zurückgreifen. Die zentralen Steuerungsfunktionen zum Erzeugen des Video-Signals nimmt bei der Hercules-Grafikkarte, wie beispielsweise auch bei der im folgenden Artikel beschriebenen CGA-Karte, der Video-Controller des Motorola-Bausteins 6845 wahr.

Der Video-Controller wandelt die Daten aus dem Bildschirmspeicher in elektrische Signale für den Monitor um. Um die im Bildschirmspeicher

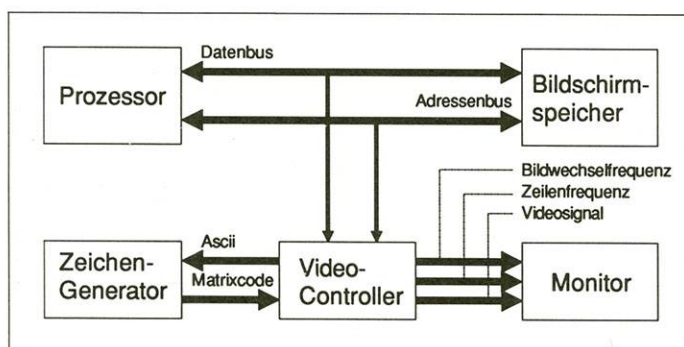


Bild 2. Der Video-Controller dient als Vermittler zwischen Prozessor, Bildschirmspeicher, Zeichengenerator und Monitor

abgelegten Daten in Punkte, Zeichen oder Grafiken auf dem Bildschirm umzuwandeln, liest der Video-Controller diese Daten ein und formt darauf die elektrischen Signale für den Monitor, der dann den gewünschten Text oder die Grafik anzeigt. Im Text-Modus interpretiert der Video-Controller die Bytes im Bildschirmspeicher als ASCII-Werte und ordnet ihnen eine Bildpunktmatrix zu, die den Aufbau des entsprechenden Zeichens auf dem Bildschirm übernimmt.

Da in diesem Fall ein einzelnes Byte mehrere Punkte auf dem Bildschirm definiert, ist der Speicherbedarf im Text-Modus natürlich wesentlich geringer als im Grafikmodus. Um den dadurch zusätzlich zur Verfügung stehenden Speicherplatz zu nutzen, arbeitet der Computer im Text-Modus mit mehreren Bildschirmseiten. Das bedeutet, daß sich im Bildschirmspeicher gleichzeitig mehrere Bilder befinden. So arbeitet beispielsweise die im nächsten Artikel vorgestellte CGA-Karte im Text-Modus bei 80 Zeichen pro Zeile mit vier Bildschirmseiten und bei 40 Zeichen pro Zeile sogar mit acht Bildschirmseiten. Der Monochromadapter verfügt allerdings nur über eine Bildschirmseite.

Zwischen der Text- und der Grafikdarstellung existieren große Unterschiede

Im Grafik-Modus muß der Video-Controller im Gegensatz zum Text-Modus die Bits des Bildschirmspeichers auslesen und sie – unter Berücksichtigung der eventuell vorhandenen Farbinformationen – direkt als Bildpunkt auf den Bildschirm umsetzen. Einfache Controller verwenden für die Zeichendarstellung im Text-Modus eine Matrix von 5x7 Punkten, so daß sich Zeichen mit Unterlängen wie die Buchstaben, g, j, p, q oder y nicht zufriedenstellend darstellen lassen. Der Video-Controller einer PC-Grafikkarte arbeitet dagegen mit einer Matrix von 8x8 Bildpunkten (CGA), 14x8 Bildpunkten (EGA und VGA) oder sogar 14x9 Bildpunkten (MDA und Hercules). Aufgrund

dessen treten bei Unterlängen keine Probleme mehr auf, und die Zeichen gewinnen mit zunehmender Bildpunktauflösung der Zeichenmatrix an Schärfe.

Natürlich ließe sich die Bildauflösung der Zeichenmatrix und damit die Wiedergabequalität der einzelnen Buchstaben erheblich steigern. Doch dann müßte man auch die Grafikauflösung der Bildschirmkarte verbessern, damit sich 80x25 Zeichen auf dem Bildschirm darstellen lassen. Dafür benötigt man einen größeren Bildschirmspeicher, der die Kosten erheblich in die Höhe treibt.

Bild 2 zeigt die Arbeitsweise des Video-Controllers im PC. Der Video-Controller kommuniziert über den Daten- und Adreßbus mit dem Prozessor und dem Bildschirmspeicher. Das heißt, daß Prozessor und Video-Controller über einen eigenen Port gleichzeitig auf den Bildschirmspeicher zugreifen können. Dadurch läßt sich das lästige »Schneetreiben« vermeiden, das entsteht, wenn der Prozessor Daten in den Bildschirmspeicher schreiben will, während der Video-Controller Daten ausliest.

Der Controller liest in periodischen Zeitabständen (50-, 60- oder 70mal in der Sekunde) den Inhalt einer Bildschirmseite im Text-Modus beziehungsweise den gesamten Bildschirmspeicher im Grafikmodus oder beim Monochrom-Adapter und formt ihn in Signale für den Monitor um. Wenn der PC im Text-Modus arbeitet, so übergibt der Controller die eingelesenen Bytes aus dem Bildschirmspeicher als ASCII-Werte an den Zeichengenerator weiter. Jeder ASCII-Wert gibt eine Adresse an, unter der im Zeichengenerator die Zeichenmatrix zu dem entsprechenden Zeichen abgelegt ist. Diese Zeichenmatrix liest der Video-Controller und erzeugt daraus über ein sehr schnelles Schieberegister das serielle Video-Signal.

Befindet sich der PC dagegen im Grafik-Modus, so kann der Video-Controller die aus dem Bildschirmspeicher eingelesenen Daten direkt über das Schieberegister in das serielle Vi-

deo-Signal umformen. Doch das Video-Signal allein reicht nicht aus, um ein Bild auf dem angeschlossenen Monitor entstehen zu lassen. Vielmehr muß der Video-Controller auch zwei Steuersignale erzeugen, die den Bildaufbau regeln. Ein Steuersignal ist die Bildwechselfrequenz, die für die vertikale Synchronisation zuständig ist. Das andere ist die Zeilenfrequenz, die die horizontale Synchronisation übernimmt. Die Bildröhre des Monitors baut das Bild aus horizontalen Zeilen auf, indem der Elektronenstrahl der Bildröhre von links nach rechts auf den Bildschirm schreibt. Beim Rücklauf des Strahls vom Zeilenende zum Anfang der nächsten Zeile wird der Strahl dunkel gestaltet beziehungsweise ausgetastet, damit er unsichtbar bleibt.

Der Monitor bringt die einzelnen Zeilen von oben nach unten auf den Bildschirm und kehrt am Ende der letzten Zeile wieder diagonal zum Anfang der ersten Zeile zurück. Auch bei diesem Rücklauf wird der Strahl ausgetastet. Diese Methode des Bildaufbaus auf dem Bildschirm ist bei allen Monitoren identisch. Ein solches Bild besteht nach deutscher Norm beim Fernsehgerät aus 625 Bildschirmzeilen und wird 25mal pro Sekunde aufgebaut. Die Trägheit des menschlichen Auges (die Flimmergrenze) liegt jedoch bei 30 bis 40 Hz. Dadurch entsteht bei einer Bildwechselfrequenz von 25 Hz beim Betrachter der Eindruck, daß das Bild flimmert.

Um diesen Effekt zu verhindern, hat man sich einen einfachen aber wirkungsvollen Trick einfallen lassen. Er besteht darin, daß man das sogenannte Vollbild in zwei Halbbilder aufteilt.

Das erste Halbbild besteht aus den Bildschirmzeilen mit den ungeraden, das zweite Halbbild aus den Bildschirmzeilen mit den geraden Zeilennummern. Der Elektronenstrahl schreibt nun zuerst die Zeilen des ersten Halbbildes und danach die Zeilen des zweiten Halbbildes auf den Schirm, so daß im Endeffekt 50mal pro Sekunde ein Halbbild erzeugt



Bild 3. Das Steuerregister des 6845 kann nur beschrieben werden



Bild 4. Aus dem Statusregister des Video-Controllers können Sie nur lesen

wird. Dieses Halbbildverfahren hat den Vorteil, daß das Bild für das menschliche Auge nicht mehr flimmert und die Video-Bandbreite nicht erhöht werden muß.

Bei preiswerten Home-Computern wendet der Video-Controller ebenfalls das Halbbildverfahren zum Bildaufbau an und speist sein auf einen hochfrequenten Träger aufmoduliertes Video-Signal in die Eingangsbuchse des Fernsehgerätes. Dieses Verfahren ist jedoch im Grunde genommen für Computer ungeeignet, weil das Computerbild oft statisch und dadurch das Flimmern trotz einer Bildwechselfrequenz von 50 Hz deutlich zu erkennen ist. Insbesondere bei Textverarbeitungs-, Datenbank- und Tabellenkalkulations-Programmen bekommt der Anwender diesen Nachteil zu spüren. Deshalb hat man sich beim PC nicht auf das unprofessionelle Halbbildverfahren eingelassen.

Die Grafikkarten am PC arbeiten nicht nach dem Halbbild-Verfahren und flimmern dadurch wesentlich weniger

Die Grafikkarten des PC stellen die Bilder mit einer echten Bildwechselfrequenz von 50 Hz (Monochromadapter und Hercules-Grafikkarte), 60 Hz (CGA- und EGA-Karte) oder sogar 70 Hz (VGA-Karte) dar. »Echte Bildwechselfrequenz« bedeutet in diesem Fall, daß die Grafikkarten 50, 60 oder 70 Vollbilder pro Sekunde aufbauen.

Zur Programmierung des Video-Controllers stellt das BIOS des PC den Video-Interrupt 10 zur Verfügung. Mit den Funktionen, die in der Tabelle des vorangegangenen Artikels abgebildet sind, können Sie die Bildschirmkarten kontrollieren. Der Monochromadapter und die CGA-Farbgrafikkarte arbeiten beide mit dem Video-Controller 6845. Dieser Video-Controller läßt sich über seinen internen Registersatz von 18 Registern (0 bis 17) direkt programmieren. Allerdings müssen Sie dabei sehr sorgfältig vorgehen und die bestehenden Parameter nur in kleinen Schritten ändern, weil sonst der Monitor Schaden nehmen kann. Die internen Register des Video-Controllers 6845 lassen sich über das Adreßregister unter der Portadresse 3B4hex (MDA) oder 3D4hex (CGA) auswählen und über das Datenregister unter der Portadresse 3B5hex (MDA) oder 3D5hex (CGA) beschreiben (nur die Register 00 bis 0Fhex) und auslesen (nur die Register 10 und 11).

Tabelle 1 zeigt den kompletten Registersatz des 6845 mit den Voreinstellungen für verschiedene Video-Modi

Der Registersatz des Video-Controllers 6845						Voreinstellung	Bedeutung
Register	Bits	CGA- Text-Modus mit 80 Zei- chen/Zeile	CGA- Text-Modus mit 40 Zei- chen/Zeile	CGA-Gra- fik-Modus	MDA		
0	0 bis 7	97	113	56	56		Zahl der (teilweise unsichtbaren) Textzeilen pro Textzeile-1 (für HSync-Signal)
1	0 bis 7	80	80	40	40		Zahl der tatsächlich angezeigten Textzeilen pro Textzeile
2	0 bis 7	82	90	45	45		Rasterzeilen-Position des HSync-Signals (bestimmt horizontale Verschiebung)
3	0 bis 3	15	10	10	10		Länge des HSync-Signals
	4 bis 7	0	0	0	0		Länge des VSync-Signals (0 = 16 Zeiteinheiten)
4	0 bis 6	25	31	31	127		Zahl der (teilweise unsichtbaren) Textzeilen-1
5	0 bis 4	6	6	6	6		Feineinstellung für Register 4 in Rasterzeilen
6	0 bis 6	25	25	25	100		Zahl der tatsächlich angezeigten Textzeilen
7	0 bis 6	25	28	28	112		Zeichenposition des VSync-Signals-1 (bestimmt senkrechte Verschiebung)
8	0 und 1	2	2	2	2		Zeilensprung-Modus (Interlace) ein (1 oder 3) oder aus (0 oder 2)
9	0 bis 4	13	7	7	1		Höhe eines Zeichens in Rasterzeilen-1
10	0 bis 3	11	6	6	6		Erste Rasterzeile des Cursors (maximal 12)
	5 und 6	0	0	0	0		Bit 6 = 0 und Bit 5 = 0: Kein Blinken
							Bit 6 = 0 und Bit 5 = 1: Kein Cursor
							Bit 6 = 1 und Bit 5 = 0: Blinken
11	0 bis 3	12	7	7	7		Bit 6 = 1 und Bit 5 = 1: Langsam blinkender Cursor
							Letzte Rasterzeile des Cursors (maximal 12)
12	0 bis 5	-	-	-	-		Bit 8 bis Bit 13 der Startadresse des Bildschirmspeichers
13	6 und 7	0	0	0	0		-
	0 bis 7	-	-	-	-		Bit 0 bis Bit 7 der Startadresse des Bildschirmspeichers
14	0 bis 5	-	-	-	-		Bit 8 bis Bit 13 der Adresse des Cursors (maximal 1999)
15	6 und 7	0	0	0	0		-
	0 bis 7	-	-	-	-		Bit 0 bis Bit 7 der Adresse des Cursors (maximal 1999)
16	0 bis 5	-	-	-	-		Bit 8 bis Bit D der Bildschirmadresse, bei der ein Lichtgriffelimpuls auftrat
17	6 und 7	0	0	0	0		-
	0 bis 7	-	-	-	-		Bit 0 bis Bit 7 der Bildschirmadresse, bei der ein Lichtgriffelimpuls auftrat

Tabelle 1. Der Registersatz des Video-Controllers bestimmt den Bildschirmaufbau

Adresse	Funktion
3B0hex	-
3B1hex	-
3B2hex	-
3B3hex	-
3B4hex	Indexregister des Video-Controllers
3B5hex	Datenregister des Video-Controllers
3B6hex	-
3B7hex	-
3B8hex	Steuerregister des Bildschirmadapters
3B9hex	Lichtgriffel-Flipflop setzen
3BAhex	Statusregister des Bildschirmadapters
3BBhex	Lichtgriffel-Flipflop löschen
3BChex	Datenregister der Druckerschnittstelle
3BDhex	Statusregister der Druckerschnittstelle
3BEhex	Steuerregister der Druckerschnittstelle
3BFhex	Konfigurationsregister (Hercules-Grafikkarte)

Tabelle 2. Der Adreßbereich eines Monochromadapters und einer Hercules-Grafikkarte

und den Bedeutungen der Register. Für »gefahrloses« Experimentieren sind die Register 10, 11, 14 und 15 geeignet. Die Register 10 und 11 geben die Start- und Endzeile des Bildschirm-Cursors an. Beim Monochromadapter lautet die Startzeile 11 und die Endzeile 12, während bei der CGA-Grafikkarte der Cursor von Zeile 6 bis 7 reicht.

In beiden Fällen stellt der Video-Controller einen zweizeiligen Cursor dar. Ändern Sie die Startzeile beispielsweise in 0 um, so erhalten Sie einen Block-Cursor. Die Register 14 und 15 bestimmen die Cursor-Position auf dem Bildschirm. Als Position fungieren die Werte 0 bis 1999, wobei 0 die linke obere Bildschirmcke und 1999 die rechte untere Bildschirmcke bezeichnet. Die Blinkfrequenz des Cursors läßt sich softwaremäßig leider nicht verändern, weil deren Steuerung durch die Hardware erfolgt. Neben den 18 internen Registern sowie dem Adreß- und Datenregister besitzt der Video-Controller unter der Portadresse 3B8hex ein Steuerregister (Bild 3) und unter der Portadresse 3BAhex ein Statusregister (Bild 4). Über Bit 3 des Steuerregisters wird beispielsweise der gesamte Bildschirm dunkel geschaltet, und Bit 5 läßt den gesamten Bildschirm blinken. Bit 0 des Statusregisters ist wichtig, wenn Sie Daten ohne Umweg über den Video-Interrupt 10 direkt in den Bildschirmspeicher schreiben möchten. Ist dieses Bit gesetzt, so ist das Signal zur horizontalen Synchronisation (Strahlrücklauf) aktiv, und Sie können ein Byte in den Bildschirmspeicher schreiben, ohne »Bildschirmschnee« durch den gleichzeitigen Speicherzugriff des Video-Controllers befürchten zu müssen.

In der Hercules-Grafikkarte können zwei Bilder untergebracht werden, zwischen denen man umschalten kann

Die Hercules-Grafikkarte besitzt gegenüber dem Monochromadapter zwei zusätzliche Register, die die Konfiguration und den Betriebsmodus der Hercules-Grafikkarte bestimmen. Über das Konfigurations-Register läßt sich der insgesamt 64 KByte große Bildschirmspeicher der Hercules-Grafikkarte in drei Schritten ein- und ausblenden. Die drei zulässigen Größen für den Bildschirmspeicher werden mit »Diag«, »Half« und »Full« bezeichnet. In der Diag-Konfiguration sind nur 4 KByte des Bildschirmspeichers aktiv. In diesem Modus verhält sich die Hercules-Grafikkarte wie ein Monochromadapter. Daher ist bei dieser Konfiguration auch nur die

Textdarstellung möglich. Die Half-Konfiguration aktiviert, wie es der Name schon andeutet, die Hälfte des Bildschirmspeichers, also 32 KByte (Adresse B0000hex bis B7FFFhex). Um die Hercules-Grafikkarte im Grafik-Modus zu betreiben, ist zumindest die Half-Konfiguration erforderlich.

Die Half-Konfiguration ist dann sinnvoll, wenn die Hercules-Grafikkarte zusammen mit einer CGA-Farbgrafikkarte in einem PC installiert ist. Anderenfalls würden sich die oberen 32 KByte der Hercules-Grafikkarte mit dem Bildschirmspeicher der CGA-Farbgrafikkarte überschneiden. Die Full-Konfiguration blendet schließlich den gesamten Bildschirmspeicher der Hercules-Grafikkarte ein (Adresse B0000hex-BFFFFhex).

Weil eine komplette Grafikseite der Hercules-Grafikkarte knapp 32 KByte des Bildschirmspeichers benötigt, erhalten Sie mit der Full-Konfiguration zwei Grafikseiten, zwischen denen Sie über Bit 7 des Modus-Registers umschalten. Tabelle 2 zeigt die Portadressen des Monochromadapters und der Hercules-Grafikkarte. Bit 0 und 1 des Konfigurations-Registers bestimmen den Aufbau des Bildschirmspeichers. Wenn Bit 0 gelöscht ist, ist es auch nicht möglich, das Grafik-Modus-Bit 1 im Steuerregister der Hercules-Grafikkarte zu setzen. Nach dem Einschalten des PC ist das Bit 0 des Konfigurationsregisters immer gelöscht.

Ist Bit 0 dagegen gesetzt, so läßt sich das Grafik-Modus-Bit setzen. Wenn Bit 1 gelöscht ist, ist die zweite Bildschirmseite der Hercules-Grafikkarte, also der Bildschirmspeicher von Adresse B8000hex bis BFFFFhex, ausgeblendet. Außerdem bleibt das Setzen des Seitenauswahl-Bit im Steuerregister der Hercules-Grafikkarte ohne Wirkung. Nach dem Einschalten des PC ist das Bit 1 des Konfigurationsregisters immer gelöscht. Ist Bit 1 dagegen gesetzt, so ist auf der Hercules-Grafikkarte der Speicherbereich von B8000hex bis BFFFFhex aktiviert. Das heißt, daß sich die zweite Grafikseite nun durch Setzen des Seitenauswahl-Bits einschalten läßt.

Die drei bereits angesprochenen Bildschirmspeicher-Konfigurationen »Diag«, »Half« und »Full« lassen sich damit durch einen einfachen Out-Befehl unter Basic auswählen. Der Befehl

```
out &h3bf,0
aktiviert die Diag-Konfiguration, mit
out &h3bf,1
wird die Half-Konfiguration eingeschaltet und
out &h3bf,3
```



WIR WÜRDEN UNSER DBÜ GERN
MIT ANDEREN PRODUKTEN
VERGLEICHEN ... FRAGT SICH
NUR WOMIT ??!

**E S G I B T
K E I N E N
A N D E R E N**

**dBASE*-COMPILER/
dBASE* NACH C-ÜBERSETZER
MIT VERGLEICHBAREN
LEISTUNGSMERKMALEN**

**DOCH
URTEILEN
SIE SELBST**



- C-QUELLCODE AUS dBASE*-CODE
- LAUFFÄHIGE EXE-FILES
- LIZENZFREIE WEITERGABE DER MIT DBÜ ERSTELLTEN EXE-FILES
- HOHE PORTABILITÄT DER ERSTELLTEN dBASE*-PROGRAMME
- UMFANGREICHER BEFEHLSVORRAT
- PROMPT/MENU/SAVE/RESTORE SCREEN
- NETZWERKFÄHIGES PROGRAMMERGEBNIS
- INTEGRIERTER C COMPILER/LINKER
- DEUTSCHE BENUTZEROBERFLÄCHE
- DEUTSCHES HANDBUCH

GEBEN SIE SICH NICHT MIT WENIGER ZUFRIEDEN!

BRINGEN AUCH SIE ÜBERRASCHENDES TEMPO
IN IHRE dBASE*-PROGRAMME! AB SOFORT
ERSTELLEN SIE C PROGRAMME, AUCH WENN
SIE DIESE PROGRAMMIERSPRACHE NICHT
BEHERRSCHEN.



DM 698,-



DEMODISKETTE/
DATENBLÄTTER

DM 10,-

WIR SUCHEN NOCH EXKLUSIV-DISTRIBUTOREN
EUROPAWEIT.

R & P GMBH

BOXGRABEN 131-133 · D-5100 AACHEN
TEL. (02 41) 40 24 07 + 40 24 71
FAX (02 41) 40 24 20

*dBASE IST EIN EINGETRAGENES WARENZEICHEN DER FIRMA ASHTON-TATE

bewirkt die Wahl der Full-Konfiguration. Über das Konfigurationsregister wird allerdings lediglich die Größe des Bildschirmspeichers bestimmt, so daß die Voraussetzungen zum Umschalten in den Grafik-Modus gegeben sind.

Um den Grafik-Modus der Hercules-Grafikkarte dann endgültig zu aktivieren, müssen Sie folgende Aktionen durchführen:

- mindestens 32 KByte Bildschirmspeicher einblenden, das heißt, die Half- oder Full-Konfiguration wählen
- das Grafik-Modus-Bit 1 im Steuerregister der Hercules-Grafikkarte setzen
- die internen Register des Video-Controllers 6845 mit neuen Werten laden
- den Bildschirmspeicher löschen

Listing 1 zeigt Ihnen unter Ausnutzung der Fähigkeiten eines Assemblers, wie man das Mode-Control-Register ausliest oder seinen Wert ändert.

Mit Hilfe des Listings 2 ermitteln Sie, ob der angeschlossene PC über eine Hercules-Grafikkarte verfügt oder nicht.

Die wichtigsten Assembler-Unterprogramme für die Erzeugung einer Grafik auf der Hercules-Grafikkarte zeigt Listing 3. Diese Prozeduren schalten die Bildschirmdatei in den Grafik-Modus und wieder zurück in den Text-Modus. Außerdem sind die Routinen zum Setzen, Löschen, Invertieren und Prüfen eines Punktes implementiert. Die Routinen müssen mit der Assembleranweisung »include <Programmname>« in das Anwendungsprogramm eingebunden

und dann mit dem Call-Befehl aufgerufen werden.

Das Demonstrationsprogramm in Listing 4 veranschaulicht die Verwendung dieser Grafikroutinen. Es erzeugt eine schräge Linie, die von links nach rechts über den Bildschirm wandert. Die vier Listings versetzen Sie in die Lage, die wichtigsten Routinen des Video-Interrupts 10 zu simulieren. Da der Interrupt 10 die Hercules-Grafikkarte nicht unterstützt, werden Sie für die Unterprogramme sicherlich Verwendung finden.

Vielleicht hätten die Entwickler von IBM bei der Veröffentlichung der EGA-Karte die Hercules-Benutzer berücksichtigen sollen. Es wäre sicherlich allen zugute gekommen.

(Markus Zietlow/sw)

```
[P30] 1: ;
[6Y2] 2: ; Name: CRT-Change
[P30] 3: ;
[DK5] 4: ; Funktion: Lesen und Ändern des
[TY4] 5: ; Mode Control Registers
[P30] 6: ;
[LP5] 7: mov ax,40h ; -Im Extra-Segment das
[325] 8: mov es,ax ; Datensegment des
[X93] 9: ; Video-BIOS ablegen
[5S6] 10: mov dx,es:[63h] ; -Port-Adresse des Index-
[NR3] 11: ; Registers ermitteln
[DZ3] 12: ; (funktioniert bei MDA,
[Y43] 13: ; Hercules und CGA)
[UN4] 14: add dl,4 ; -Steuerregister
[3P2] 15: ; spezifizieren
[EH6] 16: mov al,es:[65h] ; -Im AL-Register steht
[2K4] 17: ; der aktuelle Wert des CRT
[P30] 18: ;
[8D7] 19: mov al,neu_wert ; -Den neuen Wert in das AL-
[546] 20: out dx,al ; Register schreiben und
[KD6] 21: mov es:[65h],al ; das Register sowie den
[H73] 22: ; BIOS-Datenbereich
[VN2] 23: ; aktualisieren
```

Listing 1. Es ist kein Problem, das Mode-Control-Register auszulesen oder zu ändern

```
[P30] 1: ;
[6Y2] 2: ; Name: CRT-Change
[P30] 3: ;
[5W4] 4: ; Funktion: Ermitteln, ob eine
[N25] 5: ; Hercules-Karte vorhanden ist
[P30] 6: ;
[MU5] 7: mov ax,40h ; -Im Extra-Segment das
[736] 8: mov es,ax ; Datensegment des Video
[9G2] 9: ; -BIOS ablegen
[YH6] 10: mov dx,es:[63h] ; -Port-Adresse des Index
[DS3] 11: ; -Registers ermitteln
[DZ3] 12: ; (funktioniert bei MDA,
[0Y2] 13: ; Hercules und CGA)
[MT4] 14: add dl,6 ; -Statusregister
[8T2] 15: ; spezifizieren
[3R5] 16: mov bl,10 ; -Zähler auf 10 setzen
[924] 17: in al,dx ; -Status lesen und
[UC5] 18: and al,80h ; das Bit 7 isolieren
[8H5] 19: mov ah,al ; -AL in AH übernehmen
[HW4] 20: mov cx,8000h ; -Zählregister
[5T2] 21: ; initialisieren
[HG1] 22: schleife:
[K55] 23: in al,dx ; -Für die Dauer der
[R96] 24: and al,80h ; Zählschleife den Status
[6M4] 25: ; lesen und Bit 7 isolieren
[6C6] 26: cmp al,ah ; -Wenn sich der Inhalt von
[MG6] 27: je weiter ; Bit 7 nicht geändert hat,
[9U5] 28: dec bl ; dann Sprung zu weiter,
[ZA3] 29: ; ansonsten Zähler
```

```
[7S2] 30: ; dekrementieren
[L36] 31: jz hercules ; -Hat sich Bit 7 zehn mal
[ZV3] 32: ; geändert ==> Hercules
[XA1] 33: weiter:
[KN6] 34: loop schleife ; -Weiter in der Schleife,
[8L4] 35: ; wenn Bit 7 nicht geändert
[IA5] 36: jmp andere ; -Hat sich Bit 7 nie
[RB4] 37: ; geändert, dann ist keine
[AF4] 38: ; Hercules-Karte vorhanden
```

Listing 2. Mit Hilfe des Assembler-Programms läßt sich ermitteln, ob eine Hercules-kompatible Bildschirmdatei angeschlossen ist

```
[P30] 1: ;
[XR6] 2: ; Grafiktreiber für die Hercules-Karte
[P30] 3: ;
[W04] 4: ; DOS Extra, Ausgabe 5
[P30] 5: ;
[4X2] 6: ; Name: a.asm
[P30] 7: ;
[W56] 8: index equ 3b4h ; -Index-Register der
[RP2] 9: ; Hercules-Karte
[P96] 10: control equ 3b8h ; -Control-Register
[RB6] 11: textmod equ 28h ; -Umschalten in den
[912] 12: ; Text-Modus
[B16] 13: grafik equ 10 ; -Umschalten in den
[BB2] 14: ; Grafik-Modus
[LR5] 15: config equ 3bfh ; -Konfigurationen
[AU1] 16: ; -Schalter
[D66] 17: pagel equ 0b000h ; -Anfangssegment der
[0A1] 18: ; ersten
[HW2] 19: ; Bildschirmseite
[M86] 20: length equ 4000h ; -Länge einer Seite
[AY1] 21: ; in Worten
[2P4] 22: video equ 10h ; -Adresse des
[793] 23: ; Video-Interrupts
[P30] 24: ;
[PH6] 25: ; -Video-Controller auf Grafik schalten,
[HR6] 26: ; erste Seite anwählen und den Bildschirm
[PG1] 27: ; löschen
[P30] 28: ;
[P01] 29: graph:
[TM5] 30: mov dx,config ; -Den Full-Modus
[D35] 31: mov al,3 ; der Grafikkarte
[X64] 32: out dx,al ; anwählen
[565] 33: mov si,offset tab ; -Adresse der
[6M2] 34: ; Grafik-Tabelle
[MN3] 35: ; zur Initialisierung
[5M3] 36: cld ; freigeben
[AV5] 37: mov dl,low index ; -Adresse des 6845
[P61] 38: ; laden
[WB4] 39: mov cx,12 ; -12 Register
[Z05] 40: push cs ; -Den Inhalt des
```



```

[XL5] 41: pop ds ; CS-Registers in das
[VE3] 42: ; DS-Register bringen
[371] 43: graph1:
[9Y5] 44: lodsw ; -Initialisierungswert
t
[TU5] 45: out dx,ax ; holen und speichern
[2N5] 46: loop graph1 ; bis alle Register
[5A3] 47: ; geschrieben wurden
[2R5] 48: mov dl,low control; -Mithilfe des
[2Y5] 49: mov al,grafik ; Control-Register
[W64] 50: out dx,al ; die erste
[HW2] 51: ; Bildschirmseite
[2N1] 52: ; aufrufen
[2E5] 53: mov ax,pagel ; -Ab der Anfangs-
[L02] 54: ; adresse der
[H85] 55: mov es,ax ; Hercules-Karte
[R66] 56: mov cx,length ; den Speicher durch
[PM5] 57: xor ax,ax ; eine Exklusiv-Oder
[7C6] 58: rep stosw ; -Verknüpfung lösche
n
[3P0] 59: ret
[3P0] 60: ;
[2E7] 61: ; -Hercules-Karte in den Text-Modus schalten
[3P0] 62: ;
[2W0] 63: text:
[5N6] 64: xor ax,ax ; -Mit dem Video-Interrupt in
[VE5] 65: int video ; Text-Modus schalten
[3P0] 66: ret
[3P0] 67: ;
[A27] 68: ; -Ein Pixel auf dem Bildschirm setzen oder
[X27] 69: ; löschen (BX enthält die X-Koordinate, AX
[LW6] 70: ; enthält die Y-Koordinate, DL bestimmt,
[3K6] 71: ; ob ein Punkt gesetzt (DL=0), gelöscht
[SR5] 72: ; (DL=1) oder invertiert (DL=2) wird)
[3P0] 73: ;
[8W0] 74: plot:
[8U4] 75: call getpar ; -Bitmaske und
[3P3] 76: ; Bytenummer berechne
n
[P04] 77: or dl,dl ; -Wenn DL=0,
[YP4] 78: jnz plot1 ; dann wird der
[7U3] 79: or byte ptr es:[di],al
[CU2] 80: ; Punkt gesetzt
[3P0] 81: ret
[411] 82: plot1:
[GW3] 83: dec dl ; -Wenn DL=1,
[1R4] 84: jnz plot2 ; dann wird der
[XF4] 85: not al ; Punkt auf dem
[FX3] 86: and byte ptr es:[di],al
[PN3] 87: ; Bildschirm gelöscht
[3P0] 88: ret
[611] 89: plot2:
[S04] 90: xor byte ptr es:[di],al
[2C2] 91: ; -Anderenfalls
[LG2] 92: ; wird der Punkt
[W22] 93: ; invertiert
[3P0] 94: ret
[3P0] 95: ;
[3K6] 96: ; -Prüfen, ob ein Pixel auf dem Bildschirm
[3D7] 97: ; gesetzt ist (BX enthält die X-Koordinate,
[WC5] 98: ; AX enthält die Y-Koordinate, AL
[W46] 99: ; bestimmt, ob ein Punkt gesetzt (AL=1)
[PA4] 100: ; oder gelöscht (AL=0) ist
[3P0] 101: ;
[UA1] 102: getcol:
[8U4] 103: CALL getpar ; -Bitmaske und
[3P3] 104: ; Bytenummer berechne
n
[R44] 105: and al,byte ptr es:[di]
[RM3] 106: ; -Punkt überprüfen
[8Y5] 107: jz getcoll ; -AL=0, wenn der Punkt
t
[UR4] 108: mov al,1 ; gelöscht ist,
[0N2] 109: ; ansonsten AL=1
[E02] 110: getcoll: ret
[3P0] 111: ;
[DN6] 112: ; -Bytenummer und Bitmaske berechnen (BX
[P47] 113: ; enthält die X-Koordinate, AX enthält die
[HY6] 114: ; Y-Koordinate, in AL wird die Maske und in
[S06] 115: ; DI die Byte-Nummer zurückgegeben)
[3P0] 116: ;
[XA1] 117: getpar:
[8K4] 118: mov di,ax ; -Berechnung des
[8S2] 119: ; Zeilenoffsets
[964] 120: shr ax,1 ; mithilfe der
[L13] 121: shr ax,1 ; Formel
[VC3] 122: mov ah,90 ; Offset =
[YB2] 123: mul ah ; 90 *
[733] 124: and di,3 ; (Y div 4)
[L22] 125: ror di,1 ; +
[GS2] 126: ror di,1 ; 2000h *
[663] 127: ror di,1 ; (Y mod 4)

```

```

[E22] 128: add di, ax ;
[L76] 129: mov ax,pagel ; -ES mit der Basisadresse
[X03] 130: mov es,ax ; laden
[WZ5] 131: mov cl,bl ; -X-Koordinate für die
[A55] 132: shr bx,1 ; Bitmaske festlegen
[UG4] 133: shr bx,1 ; und Byte-Offset
[CM3] 134: shr bx,1 ; bestimmen
[W85] 135: add di,bx ; -Adresse in DI ablegen
[KA5] 136: mov al,128 ; -Bitmaske berechnen
[U53] 137: and cl,7 ; und in CL
[S93] 138: shr al,cl ; ablegen
[3P0] 139: ret
[3P0] 140: ;
[XK5] 141: ; -Tabelle für die Initialisierung
[K05] 142: ; der Grafik im Hercules-Modus
[3P0] 143: ;
[DN0] 144: tab:
[1H5] 145: dw 3500h, 2d01h, 2e02h, 0703h, 5b04h, 0205h
[MH5] 146: dw 5706h, 5707h, 0208h, 0309h, 000ah, 000bh

```

Listing 3. Die Grafik-Bibliothek bietet Ihnen mehrere Funktionen zur Ansteuerung der Hercules-Grafikkarte

```

[3P0] 1: ;
[YX6] 2: ; -Beispielprogramm zur Demonstration der
[A87] 3: ; Programmroutinen der Hercules-Bibliothek
[3P0] 4: ;
[XF5] 5: code segment ; -Programmstart
[DE5] 6: org 0100h ; -Startadresse für eine
[YT1] 7: ; COM-Datei
[DN5] 8: assume cs:code, ds:code, es:code
[3P0] 9: ;
[723] 10: beispiel proc near
[3P0] 11: ;
[4T5] 12: start: call graph ; -Aktivieren des
[T42] 13: ; Grafikmodus
[9D4] 14: xor ax,ax ; -Zähler auf 0
[U93] 15: xor bx,bx ; setzen
[YD4] 16: call linie ; -Linie zeichnen
[A60] 17: start1: inc bx ; -Linie um einen Punkt
[1N5] 18: call linie ; nach rechts versetzen
[CA4] 19: dec bx ; -Vorherige Linie
[KG3] 20: call linie ; löschen
[M85] 21: inc bx ; -Vorgang wiederholen,
[MV3] 22: cmp bx,719 ; bis das Ende
[S74] 23: jnz start1 ; erreicht ist
[0G4] 24: mov ah,0 ; -Auf Tastendruck
[4W2] 25: int 16h ; warten
[4G5] 26: call text ; -Textmodus einstellen
[7C4] 27: int 20h ; -Programm beenden
[3P0] 28: ;
[0C2] 29: ; -Linie ziehen
[3P0] 30: ;
[XH5] 31: linie: push ax ; -Register AX und
[5H3] 32: push bx ; BX retten
[HG1] 33: schleife:
[EY4] 34: mov dl,2 ; -Invertierte Linie
[0P1] 35: ; zeichnen
[RF4] 36: push ax ; -Register AX und
[5H3] 37: push bx ; BX retten
[R24] 38: call plot ; -Punkt setzen
[SD4] 39: pop bx ; -Register BX und
[495] 40: pop ax ; AX wiederherstellen
[W95] 41: inc bx ; -Diagonal um ein Pixel
[7G4] 42: inc ax ; nach rechts unten
[E61] 43: ; gehen
[0G5] 44: cmp ax,347 ; -Vorgang wiederholen,
[E72] 45: jnz schleife ; bis der untere Rand
[W93] 46: ; erreicht ist
[D82] 47: ;
[SD4] 48: pop bx ; -Register BX und
[495] 49: pop ax ; AX wiederherstellen
[3P0] 50: ret
[7B3] 51: include bkart2-3.asm
[WL3] 52: ; -Routinen-Bibliothek
[PW1] 53: ; einbinden
[7Z5] 54: beispiel endp ; -Ende der Prozedur
[NH5] 55: code ends ; -Ende des Programms
[H44] 56: end start ; -Festlegen der
[TE2] 57: ; Startadresse
[1W4] 58: end ; -Ende des Quelltextes
[000] 59: ;
[60] 60: ds ; -Ende des Programms
[HA4] 61: end start ; -Festlegen der Startad
[TF6] 61: resse
[4M0] 62: end

```

Listing 4. Das Beispiel demonstriert die Routinen der Bibliothek anhand einer Linie, die über den Bildschirm wandert

Die bunte Welt der CGA

Für die Arbeit im Text-Modus hatte IBM mit dem Monochromadapter eine sehr funktionelle Bildschirmkarte vorgestellt. Die MDA-Karte konnte jedoch das Feld der grafikorientierten Anwendungen nicht abdecken. Eine farbige Darstellung fehlte völlig. Mit der CGA-Karte schaffte IBM Abhilfe.

Als IBM 1981 den ersten PC vorstellte, bot die Firma neben dem Monochromadapter die CGA-Karte (»Color Graphics Adapter« oder »Farbgrafikadapter«) an. Die CGA-Karte unterstützt sowohl einen RGB- (einen Monitor mit getrennten Signalen für die Grundfarben Rot, Grün und Blau) als auch einen Composite-Monitor. Die Bildschirmkarte kann neben alphanumerischen Zeichen auch grafische Informationen auf der Basis eines Punkte-Systems ausgeben. Der Bildschirmadapter bildet alphanumerische Zeichen in einem 80x25 Zeichen großen Koordinatensystem ab. Die Qualität der Bildschirmdarstellung ist nicht besonders gut, da nur eine Auflösung von 640x200 Punkten erreicht wird. Die Vorteile der CGA-Karte liegen in einem völlig anderen Bereich. Für viele Anwendungen ist es sinnvoll und manchmal sogar notwendig, daß zusätzlich zum Text Diagramme, Kurven und andere grafische Informationen angezeigt werden. Farbliche Darstellungen erweitern das Aufgabengebiet und die Effizienz der Computergrafik wesentlich.

Ein Bit kontrolliert die Intensität und verdoppelt so die Zahl der Farben

Die CGA-Karte kann 16 Farben gleichzeitig darstellen. Das Signal zum Monitor setzt sich aus vier Komponenten zusammen, wobei jede Komponente entweder in das Signal eingemischt oder aus dem Signal weggelassen wird. Vier Komponenten mit je zwei möglichen Zuständen ergeben 16 Signale, die jeweils eine Farbe wiedergeben. Drei der vier Komponenten sind die Farbanteile Rot, Blau und Grün. Der vierte Signalanteil heißt »Intensität« und läßt sich mit der Helligkeit eines Punktes vergleichen.

Wenn das Intensitäts-Signal gesetzt ist, dann wird die Grundfarbe in einem helleren Ton wiedergegeben. Die Farben können deshalb in die zwei Gruppen »hell« und »dunkel« eingeteilt werden. Die Farbzusammenstellung ist in Tabelle 1 vorgestellt. Wie die Farben tatsächlich aussehen, hängt stark vom verwendeten Monitor ab. Manche Monitore können nämlich mit dem Wert für Intensität überhaupt nichts anfangen. Das bedeutet, daß sich in diesem Fall die dunkle und die helle Version einer Farbe nicht unterscheiden. Andere Monitore haben wiederum Schwierigkeiten mit der Darstellung bestimmter Farben (in den meisten Fällen Braun, Gelb und Grau). Im Text-Modus arbeitet auch die CGA-Karte so, daß die ASCII-Codes und die zugehörigen Attribute im Bildschirmspeicher stehen und von der Bildschirmkarte in Zeichenmatrizen umgewandelt werden. Auf dieses Verfahren soll im folgenden nicht eingegangen werden, da es im Zusammenhang mit den Erklärungen zum Monochromadapter und zur Hercules-Karte ausführlich beschrieben wird. In den verschiedenen Grafik-Modi werden die Punkte in Form ver-

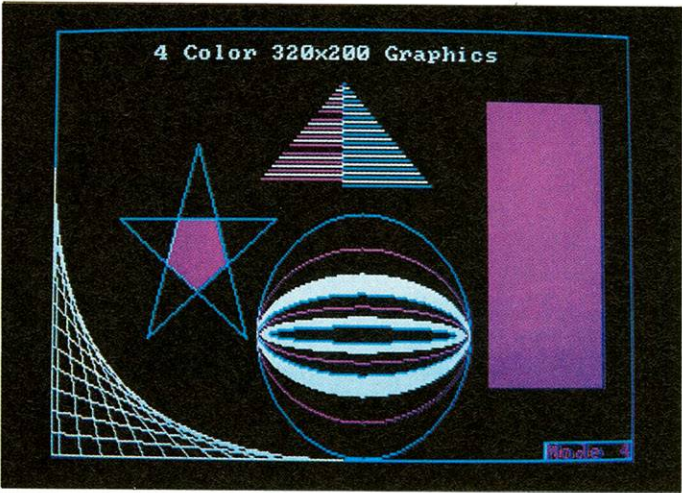


Bild 1. Die CGA-Karte brachte als erste farbige Grafiken ins Spiel

schlüsselter Signale in den Speicher geschrieben. Die Grafikkarte stellt das Farbsignal nach dem Inhalt des Bildschirmspeichers zusammen. Eine Umwandlung ist nicht notwendig. Der Bildschirmspeicher der CGA-Karte bewegt sich im Bereich von B800:0000hex bis B800:3FFFhex und ist damit 16 KByte groß. Im Unterschied zum Text-Modus ist der Bildschirmspeicher im Grafik-Modus völlig unterschiedlich aufgeteilt. Diesen Unterschied erkennt man auch im logischen Aufbau des Speichers, dessen Struktur in Bild 2 dargestellt ist.

Der Aufbau im Grafik-Modus wirkt eigenartig – eins, drei, fünf und zwei, vier, sechs wird hier gezählt

Auf den ersten Blick sieht der Speicher einfach strukturiert aus, denn die ersten 80 Byte enthalten die erste Bildschirmzeile (Zeile Nummer 0). Dann jedoch wird es interessant. Die zweite Zeile (Zeile Nummer 1) steht nämlich nicht wie in den Text-Modi direkt nach der ersten, sondern am Anfang der zweiten Hälfte des Bildschirmspeichers (ab Adresse B800:2000hex). Was aber folgt auf

Die CGA-Farbzusammenstellung						
Farbanteile			Intensität dunkel		Intensität hell	
Rot	Grün	Blau	Nummer	Farbe	Nummer	Farbe
0	0	0	0	Schwarz	8	Dunkelgrau
0	0	1	1	Blau	9	Hellblau
0	1	0	2	Grün	A	Hellgrün
0	1	1	3	Türkis	B	Helltürkis
1	0	0	4	Rot	C	Hellrot
1	0	1	5	Violett	D	Lila
1	1	0	6	Braun	E	Gelb
1	1	1	7	Hellgrau	F	Weiß

Tabelle 1. Die verschiedenen Farben der CGA-Karte

Anfangsadresse	Bildschirmspeicher	Endadresse
B8000	Zeile 0	B804F
B8050	Zeile 2	B809F
B80A0		B80EF
B9EF0	Zeile 198	B9F3F
BA000	Zeile 1	BA04F
BA050	Zeile 3	BA09F
BA0A0		BA0EF
BFEF0	Zeile 199	BFF3F

Bild 2. In den Grafik-Modi wird der Bildschirmspeicher hardwareorientiert verwaltet

die erste Zeile? Die dritte! Und nach der zweiten steht die vierte und nach der dritten die fünfte und so weiter, bis alle 200 Zeilen aufgeteilt sind. Zusammenfassend formuliert, ist der Bildschirmspeicher in den Grafik-Modi in zwei Hälften eingeteilt, die zusammen ein vollständiges Bild ergeben. Die erste Hälfte beinhaltet hintereinander alle geraden Zeilen (Nummer 0, 2, 4 und so weiter) und die zweite Hälfte alle ungeraden Zei-

len (Nummer 1, 3, 5 und so weiter). Die Frage, aus welchem Grund der Speicher in dieser Weise aufgeteilt wurde, ist leicht zu beantworten. Der Hardware-Aufbau gestaltet sich durch dieses Verfahren wesentlich einfacher. Daß die Programmierung der Bildschirmkarte dadurch entschieden schwieriger und einige Programme wesentlich langsamer werden, wurde von den Entwicklern in Kauf genommen.

Die Struktur des Bildschirmspeichers ist in allen Grafik-Modi gleich. Außerdem hat eine Zeile immer die Länge von 80 Byte. Um die Anfangsadresse einer Zeile zu bestimmen, benutzen Sie die Formel

$$\text{adresse} = (\text{zeile MOD } 2) * 8192 + (\text{INT}(\text{Zeile} / 2) * 80)$$

Da die anderen Eigenschaften des Speichers und die Verwendung der BIOS-Routinen stark vom jeweils ausgewählten Grafik-Modus abhängen, werden zunächst die Modi 4 und 5 betrachtet (320x200 Bildpunkte in vier Farben beziehungsweise Graustufen), die in Bild 2 dargestellt sind. Anschließend wird Modus 6 vorgestellt.

Der Vierfarb-Modus der CGA-Karte wird sehr häufig verwendet

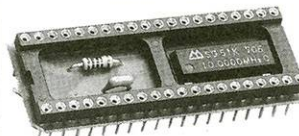
Der vierfarbige Modus 4 der CGA-Karte (Bild 1) ist der beliebteste und meistverwendete Grafik-Modus des PC. Dieser Modus bietet eine relativ hohe Auflösung in mehreren Farben; er findet deshalb in vielen Bereichen Einsatz.

Ordnung und Übersicht schaffen die beliebten DMV Sammelmappen



Bitte Bestellkarte benutzen
DMV Verlag · Postfach 250 · 3440 Eschwege

Beschleunigen Sie Ihren AT!



In allen AT-kompatiblen Rechnern läuft der Coprozessor nur mit 2/3 der Frequenz des Hauptprozessors. Daher können Sie nie die volle Leistungsfähigkeit Ihres Coprozessors 80287 nutzen.

Das können Sie nun ändern:

Unser **TURBO-SOCKEL** liefert die maximale mögliche Frequenz für Ihren 80287.

Der **TURBO-SOCKEL** ist lieferbar in 8 MHz, 10 MHz und 12 MHz.

Preis: 57,- DM

Der Einbau ist denkbar einfach:

Sie setzen den 80287 in die vergoldete Präzisionsfassung des TURBO-SOCKELS und die gesamte Einheit in die vorgesehene Fassung Ihres AT ein.

Eine Softwareanpassung ist nicht erforderlich!



Gude Analog- und Digital-systeme GmbH

Eintrachtstraße 113, 5000 Köln 1
 Telefon (0221) 13 67 35

AB-COMPUTER

IHR Laufwerk-Computerspezialist in Köln

Wir bieten Ihnen noch Beratung und Service **Wildenburgstr. 21**

5000 Köln 41 (Sülz)
 Tel. 02 21/4 30 14 42, FAX 46 65 15

Mommensstr. 72
 Ecke Gleulerstr. Ladenlokal

Floppy	2MB 1.44 formatiert PS 2 kompatibel 3.5 Zoll im 5.25 Rahmen komplett NEC 1137 H für AT	320,-
Seagate	40 MB Festplatte ST 251 40Ms o.Contr.	799,-
Panatek	AT 640 KB 14 Zoll Monitor amber 12 MHz, 1.2 MB Lw., Ser./Par. Uhr FDD/HDD Contr. AT Tast. mit FTZ Frequenz Anzeige NEC Laufwerk Cmos Kalltechnik neuestes Bios	2760,-
Schneider	Tower AT 1x3.5 Zoll Lw. Monitor Tastat.	2498,-
EIZO	Farbmonitor 8060S 28 dot. 820x620 Multisync	1548,-
EIZO	Farbmonitor 9070s 1280x768 16 Zoll mit VGA Karte 1024x768 Orchid mit Treiberdisketten Spitzenqualität	2750,-
VGA	Karte Orchid 1024x768 Ega/VGA 512KB RAM mit Treiberdisketten	850,-
NEC P6	Plus Drucker NEU 250Z.Sek. Traktor 80 KB Buffer	1698,-
STAR	LC 10 Drucker mit Traktor komplett 9 Nadeln	640,-
Panasonic	Drucker 1081 9 Nadeln 120 Z./Sek. Traktor	510,-
Witchpen	mal 5	550,-
MS-DOS	4.0	400,-
Maus	PC	89,-

Monitore, Controller, EGA, Vga Karten, Streamer, Wechselplatten sowie Software und andere Produkte in unserem INFO.

Branchenlösungen nach Wunsch/Systemberatung, Softwareerstellung, Firmenanpassungen

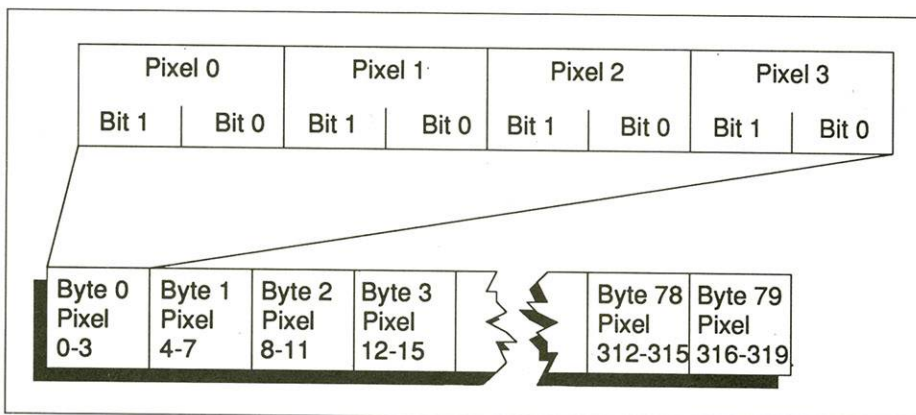


Bild 3. Im vierfarbigen Modus hat jeder Bildpunkt zwei Bits

Wie eine Zeile in diesem Modus gespeichert wird, zeigt Bild 3. Jeder Punkt auf dem Bildschirm wird in zwei Bit abgebildet, deren Wert die Farbe des Punktes bestimmt. Deshalb stehen in diesem Modus vier Farben zur Verfügung.

Es werden genau vier Grafikpunkte in jedem Byte gespeichert: die linken vier Punkte einer Zeile im ersten Byte, die nächsten vier im zweiten Byte und so weiter. Bild 3 weist auf, wie die vier Punkte (Pixel) innerhalb eines Bytes aufgeteilt sind. Damit lässt sich die Formel für die Adresse einer Grafikzeile erweitern, um die genaue Position eines Punktes zurückzugeben. An dieser Stelle reicht die Byte-Adresse nicht aus; es ist die zusätzliche Information nötig, in welchen beiden Bits der Punkt gespeichert ist.

Diese Information berechnet sich als ein Byte, in dem genau zwei Bits gesetzt sind, und zwar die zwei, die den gesuchten Punkt abbilden. Dieses Byte wird »Maske« genannt. Die Formeln lauten:

$\text{adresse} = (\text{Zeile} \text{ MOD } 2) * 8192 + (\text{INT}(\text{Zeile} / 2) * 80) + \text{INT}(\text{Spalte} / 4)$
 $\text{maske} = 192 / (4 * (\text{Spalte} \text{ MOD } 4))$

Mit diesen Informationen lassen sich leicht ein paar Programmzeilen

schreiben, die einen direkten Zugriff auf den Bildschirmspeicher erlauben. Außerdem gibt es auch BIOS-Routinen, um Bildpunkte zu lesen und zu schreiben. Eine schnelle Rechnung zeigt, daß der Speicher völlig ausgenutzt wird. 320 mal 200 Bildpunkte ergibt 64000 Bildpunkte. Geteilt durch 4 Punkte pro Byte ergibt diese Zahl 16000 Byte, um ein Bild zu speichern. Von den 16 KByte der CGA-Karte sind noch 384 Byte übrig, die nicht verwendet werden. Sie wissen nun, daß jeder Punkt in zwei Bits gespeichert wird, und wie Sie die Bits im Bildschirmspeicher finden können. Wie aber bestimmen Sie die Farbe des Punktes, und welche vier Farben der sechzehn möglichen werden verwendet?

Die Farbpalette bestimmt das Aussehen des Bildes auf dem Monitor

Diese Frage beantwortet ein neues Konzept: die Farbpalette. Die Palette beinhaltet eine Auswahl der zur Verfügung stehenden Farben. Bei den Original-IBM-Farbgrafikkarten gibt es zwei Paletten, die Paletten 0 und 1 aus Tabelle 2. Inzwischen können fast alle anderen CGA-Karten mit einer von vier Paletten arbeiten,

wobei die beiden neuen (Paletten 2 und 3 in der Tabelle 2) Versionen der Original-Paletten mit gesetztem Intensitäts-Bit sind.

Eine weitere Eigenschaft aller Paletten stellt die Hintergrundfarbe dar. Diese Farbe hat immer die Nummer 0, wobei aber jede beliebige Farbe als Hintergrundfarbe deklariert werden kann. Das heißt, in jeder Palette sind drei Farben festgelegt, während die vierte frei wählbar ist. Diese Farbe ist aber stets die Hintergrundfarbe, die auch zum Löschen des Bildschirms benutzt wird. Auch zum Setzen der Farbpalette gibt es eine BIOS-Routine im Betriebssystem.

Der letzte Modus der CGA-Karte ist der hochauflösende Modus 6. Hier besteht jede Zeile aus 640 Punkten, dem Doppelten der Modi 4 und 5. In diesem Modus wird jeder Punkt als ein Bit gespeichert, was die gleiche Ausnutzung des Bildschirmspeichers ergibt wie bei den beiden anderen Grafik-Modi. Der Aufbau einer Zeile ähnelt im übrigen den Modi 4 und 5 und ist in Bild 4 zu sehen. Die Zeilenadressen sind sogar identisch, da eine Zeile genau so viele Bits benötigt wie vorher. Die Maske beansprucht jedoch nur ein einziges Bit.

Die beiden Formeln für die Byte-Adresse und die Bit-Maske eines Punktes in Modus 6 lauten:

$\text{adresse} = (\text{Zeile} \text{ MOD } 2) * 8192 + (\text{INT}(\text{Zeile} / 2) * 80) + \text{INT}(\text{Spalte} / 8)$
 $\text{maske} = 128 / (2 * (\text{Spalte} \text{ MOD } 8))$

Um einen Punkt zu lesen oder zu schreiben, greifen Sie wiederum entweder auf das BIOS oder direkt auf den Bildschirmspeicher der Grafikkarte zu.

Da es nur einen Punkt pro Bit gibt, existieren auch nur zwei Farben in diesem Modus. Die Original-IBM-CGA-Karte konnte nur Schwarz als Hintergrund- und Weiß als Vordergrundfarbe im Modus 6 darstellen. Inzwischen ist es aber üblich, daß die Vordergrundfarbe frei ausgewählt werden kann.

(Markus Zietlow/sw)

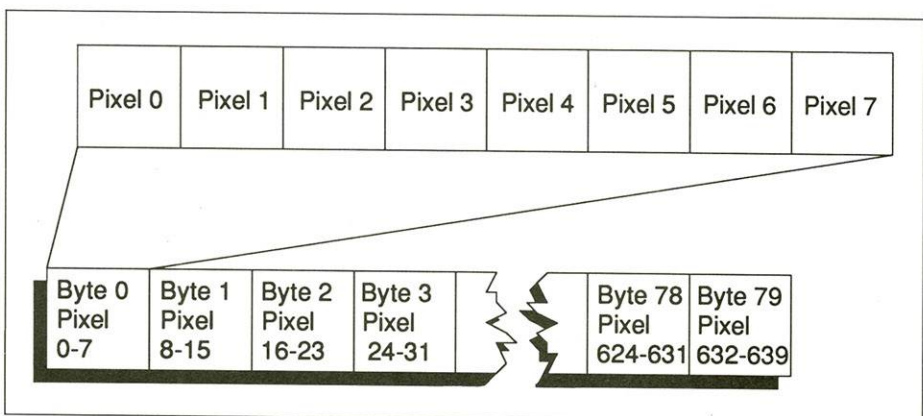


Bild 4. In der höchsten Auflösung entspricht jedes Bit im Bildschirmspeicher einem Bildpunkt

Palette	Farbe 0	Farbe 1	Farbe 2	Farbe 3
0	(*)	Grün	Rot	Braun
1	(*)	Türkis	Magenta	Hellgrau
2	(*)	Hellgrün	Hellrot	Gelb
3	(*)	Helltürkis	Lila	Weiß

(*) Farbe 0 ist standardmäßig Schwarz, kann aber auf jede beliebige Farbe gesetzt werden

Tabelle 2. Die Farbpaletten der CGA-Karte

EGA:

Farbenvielfalt wirkt elegant

Vor der EGA-Karte hatte IBM schon zwei Bildschirmkarten vorgestellt (MDA und CGA). Hercules vermarktet seine hochauflösende, grafikfähige Monochromkarte überaus erfolgreich. Doch das Ergebnis war vor allem für kommerzielle Anwendungen nicht zufriedenstellend. Es fehlte eine hochauflösende Farbgrafikkarte.

Da der Color Graphics Adapter (CGA) die Aufgaben einer qualitativ ansprechenden Farbgrafikqualität nicht erfüllen kann, wurde bald nach seiner Einführung der Ruf nach einer hochauflösenden, farbfähigen Bildschirmkarte laut. Um die Nachfrage zu befriedigen, bot IBM ab Anfang 1985 die EGA-Karte (»Enhanced Graphics Adapter« oder »Erweiterter Grafikadapter«) an. Sie bietet eine höhere Auflösung als die CGA-Karte. Die EGA-Karte »beherrscht« alle Betriebs-Modi des Monochromadapters und der CGA-Karte. Im Gegensatz zur MDA-Karte ist die EGA-Karte auch auf monochromen Bildschirmen grafikfähig. Bei Einsatz von Farbbildschirmen ist sie, im Gegensatz zu der CGA-Karte, in der Lage, 16-farbige Textbildschirme und Grafiken in einer Auflösung von 640x350 Punkten auf den Bildschirm zu bringen.

Mit der EGA-Karte wurden endlich 16 aus 64 Farben darstellbar

Ein weiterer Vorteil der EGA-Karte liegt in der Qualität des Monitorbildes. Aufgrund der preiswerten EGA-kompatiblen Bildschirmkarten hat sich so der Enhanced Graphics Adapter zum heutigen Grafikstandard für Personal Computer entwickelt. Insgesamt 16 Farben aus einer Farbpalette von 64 Farben der EGA-Karte können gleichzeitig auf einem EGA-Monitor abgebildet werden. Der Hauptunterschied zu den vorher veröffentlichten Bildschirmkarten zeigt sich darin, daß Programme, die direkt in die Register des Video-Controllers 6845 schreiben, nicht auf einer EGA-Karte lauffähig sind. Die Elektronik einer EGA-Karte ist so komplex, daß die Leistungen des 6845-Bausteins nicht ausreichen. Aus diesem Grund sind EGA-Karten mit modernen, hochintegrierten Gate Ar-

rays bestückt, die speziell an die Funktionen der EGA-Karte angepaßt wurden.

Die Zeichenmatrix der EGA-Karte besteht wie bei der Hercules-Grafikkarte aus 8x14 Bildpunkten

Die EGA-Karte arbeitet gewöhnlich mit einer Zeichenmatrix von 8x14 Bildpunkten. Damit ist sie in der Zeichen-Auflösung einer Hercules-Grafikkarte ebenbürtig. Die Hercules-Grafikkarte arbeitet zwar mit einer Matrix von 9x14 Bildpunkten, jedoch ist hier die erste Matrixspalte grundsätzlich unbenutzt, so daß sich ebenfalls eine effektive Auflösung von 8x14 Bildpunkten pro Zeichen ergibt. Die CGA-Farbgrafikkarte kann mit einer Auflösung von 8x8 Bildpunkten pro Zeichen nicht mithalten.

Bei der EGA-Karte lassen sich aufgrund der Gesamtauflösung von 640x350 Bildpunkten mit dem alternativen Zeichensatz nicht nur die üblichen 80x25 Zeichen, sondern sogar 80x43 Zeichen gleichzeitig auf dem Bildschirm darstellen. Das bedeutet einen Zugewinn von 18 Textzeilen und bringt vor allem bei Textverarbeitungs- und Tabellenkalkulationsprogrammen einen bedeutenden Vorteil.

Die EGA-Karte besitzt einen Bildschirmspeicher von 256 KByte Länge (die ersten EGA-Karten hatten allerdings nur 64 KByte). Der Bildschirmspeicher ist »dual-ported«, das heißt, daß sowohl der Prozessor als auch die Grafikkarte auf diesen Speicherbereich zugreifen können. Der Bildschirmspeicher beginnt an Adresse A0000hex und reicht bis zur Adresse BFFFFhex. Wenn ein Programm jedoch die CGA-Video-Modi 4 bis 6 anspricht, werden die Grafikdaten aus Kompatibilitätsgründen ab der Spei-

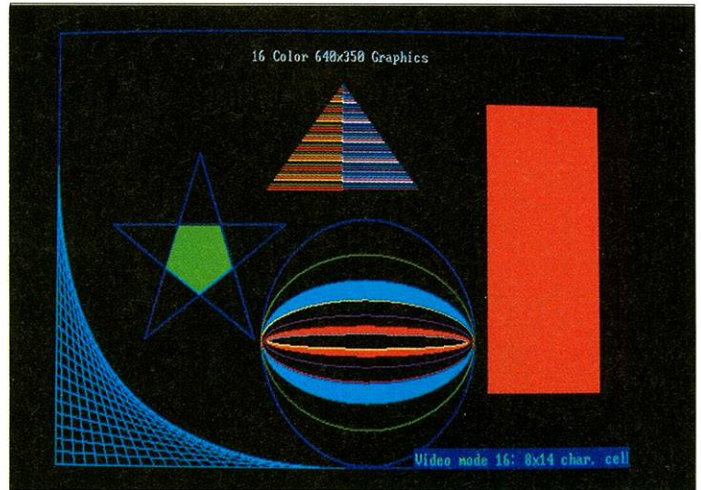
cheradresse B8000hex abgelegt. Bei der Auswahl des Video-Modus 7 beginnt der Bildschirmspeicher aus dem gleichen Grund an der Adresse B0000hex.

Wählt ein Programm die CGA-Textmodi 0 bis 3 aus, dann gibt die EGA-Karte alle Zeichen automatisch in der höheren Auflösung von 320x350 Bildpunkten beziehungsweise 640x350 Bildpunkten aus. Diese trägt entscheidend zur Lesbarkeit der Texte bei. Die Definition der Rahmenfarbe (Tabelle 2) ist bei der EGA-Karte im Gegensatz zur CGA-Farbgrafikkarte nur noch in den Grafik-Modi 4 bis 6 und in den erweiterten Modi 13 und 14 möglich. Das Farbattribut sehen Sie in Tabelle 1.

Mit 256 KByte Speicher kann die EGA-Karte voll auftrumpfen: eigene Zeichensätze und Softscrolling

In den Text-Modi 0 bis 3 und den hochauflösenden Modi 15 und 16 wird der Rahmen grundsätzlich in der Farbe dargestellt, die im Rahmenfarbregister angegeben ist. Ein 16farbiges EGA-Bild in einer Auflösung von 640 mal 350 Bildpunkten belegt 896000 (640x350x4) Bit – beziehungsweise ungefähr 110 KByte Speicher.

Wozu werden die übrigen 146 KByte der EGA-Karte verwendet? Der freie Speicherbereich einer EGA-Karte kann für die Definition von bis zu vier Zeichensätzen à 256 Zeichen (maximale Zeichenmatrix: 32x8 Bildpunkte) und für sogenanntes »weiches Scrolling« genutzt werden. Bei dieser Art von Scrolling wird nicht der Bildinhalt im Bildschirmspeicher verschoben, sondern die Grenzen für das Bild innerhalb des Speichers. Dadurch muß man der Video-Logik lediglich die neue Start-



Die EGA-Karte erlaubte erstmals farbige Grafiken in einer hohen Qualität



Das Buch zu BASIC2

PC1512/1640-Besitzer können aufatmen. Jetzt ist sie da, die BASIC2-Toolbox. Mit diesem einzigartigen Buch sind Sie nun in der Lage, das Locomotive BASIC2 besser auszunutzen, die Programmierung wird zum Kinderspiel.

Die BASIC2-Toolbox bietet nicht die x-te Einführung, sondern gibt anhand leistungsstarker und praxisorientierter Routinen und Programme den nötigen Durchblick.

Schritt für Schritt lernen Sie die strukturierte Programmentwicklung bis hin zum fertigen und lauffähigen Programm. Der Text ist in leicht und für jedermann verständlicher Form geschrieben, der Lerneffekt ist quasi garantiert. Alle Beispielpprogramme sind sofort nachvollziehbar und stammen aus der täglichen Anwendungspraxis.

Einige Beispiele aus dem Inhalt:

Kurze Einführung in die Software-Entwicklung
– Grundlagen und Struktogramme

Werkzeuge für BASIC2

- Lister ermöglicht die formatierte Druckausgabe für BASIC2-Programme
- COMP komprimiert Ihre BASIC2-Programme
- Preprozessor für Include-Dateien
- Cross ist ein Generator zur Erzeugung von Querverweislisten

Werkzeuge zur Behandlung von Textdateien

- Auswertung von Word-Textdateien
- CUT, ein Filter für Textdateien
- PASTE, Vereinigung von Textdateien

Werkzeuge zur Software-Entwicklung

- CALC, ein Rechner für verschiedene Zahlensysteme
- DUMP, Ausgabe beliebiger Dateien im Hexformat.
- COMHEX, Umwandlung von COM-Dateien in HEX-Files

Ein kompletter Disassembler für den Intel 8086

- Von der Struktur zum fertigen Programm.
- Ein Disassembler wird programmiert.

Die BASIC2-Toolbox

Autor: Günter Born,
250 Seiten, 54 Abb.,

ISBN Nr. 3-926177-01-2 (Best.-Nr. 402)

Inland:		Ausland:	
Einzelpreis	49,- DM	Einzelpreis	49,- DM
zzgl. Versandkosten	3,- DM	zzgl. Versandkosten	5,- DM
Endpreis	52,- DM	Endpreis	54,- DM

Zu beziehen über den Computerfachhandel, den guten Fachbuchhandel oder direkt beim Verlag. Händleranfragen erwünscht.

Bitte benutzen Sie unsere Bestellkarte

DMV-Verlag

Postfach 250 · 3440 Eschwege

Grafik

und Endadresse des Bildes im Speicher übermitteln, während im anderen Fall aufwendige Speicherverschiebungen notwendig wären. Das weiche Scrolling ist folglich wesentlich schneller als das konventionelle Scrolling der vorangegangenen Bildschirmkarten. Bemerkenswert ist, daß der Textbildschirm folglich Punkt für Punkt bewegt werden kann, und sich somit vollkommen ruckfrei bewegt.

Über die Speicherstelle 0:410hex läßt sich feststellen, welche Art von Monitor zur Zeit aktiv ist. Wenn die Speicherzelle den Wert 2Dhex beinhaltet, handelt es sich um einen Farbmonitor. Ein Inhalt von 3D weist auf einen Monochrom-Monitor hin.

Die Speicherstelle 0:488hex gibt darüber Auskunft, ob es sich bei dem Farbmonitor um einen CGA-Monitor, einen EGA-Monitor oder einen Monochrom-Monitor handelt. Ein Speicherzelleninhalt von F8hex steht stellvertretend für einen CGA-Monitor, F9hex für einen EGA-Monitor und FBhex für einen Monochrom-Monitor.

Über die Speicherstelle 0:4A8hex läßt sich schließlich überprüfen, ob im PC überhaupt eine EGA-Karte vorhanden ist. Wenn 0:4A8hex einen Wert beinhaltet, der größer als 0 ist, dann ist eine EGA-Karte angeschlossen, andernfalls nicht.

Der Interrupt 10 hat auch bei der EGA-Karte alles unter Kontrolle

In Maschinensprache sprechen Sie die EGA-Karte, wie den Monochromadapter beziehungsweise die CGA-Farbgrafikkarte, über den Interrupt 10 an, dessen Funktionen Sie dem ersten Teil der Grafikkarten-Serie in diesem Sonderheft entnehmen können.

Achten Sie bitte darauf, das BP-Register vor dem Aufruf einer EGA-Funktion in Maschinensprache zu retten, denn die erweiterten und neuen EGA-Funktionen des Interrupt 10 verwenden im Gegensatz zu den alten Funktionen (0 bis 15) dieses Registers und verändern dementsprechend dessen Inhalt.

Für die Darstellung von grafischen Daten und Bit-Mustern ist der Bildschirmspeicher der EGA-Karte in mehrere gleichgroße Bereiche (die sogenannten »Planes«) aufgeteilt. Es existieren insgesamt vier Planes, die von 0 bis 3 durchnummeriert sind. Die Größe der einzelnen Planes hängt mit der Größe des Bildschirmspeichers direkt zusammen. In der voll ausgebauten EGA-Karte mit 256 KByte ent-

Der Aufbau des Farbattributs im Grafik-Modus

Byte	Farbe
0	Schwarz
1	Blau
2	Grün
3	Kobaltblau
4	Rot
5	Violett
6	Braun
7	Weiß
8	Grau
9	Hellblau
A	Hellgrün
B	Hellkobaltblau
C	Hellrot
D	Hellviolett
E	Gelb
F	Hellweiß

Tabelle 1. Im Grafik-Modus bestimmen die Bits 0 bis 3 (Rot, Grün, Blau und Intensität) den Aufbau des Farbattributs

Der Aufbau des Farbwertes für ein Paletten- oder Rahmenfarb-Register

Bit	Aufgabe
0	Erstes Blau
1	Erstes Grün
2	Erstes Rot
3	Zweites Blau
4	Zweites Grün
5	Zweites Rot
6	–
7	–

Tabelle 2. Mit den 2x3 Farb-Bits des Farbwertes lassen sich 64 verschiedene Farben definieren

Aufbau der Planes

Bezeichnung	Größe
Bank 0	64 KByte
Bank 1	64 KByte
Bank 2	64 KByte
Bank 3	64 KByte

Tabelle 3. Bis zu vier Speicherbänke übernehmen die Verwaltung der Bit-Planes

halten die Planes jeweils 64 KByte. Bei der älteren 64-KByte-Karte ist die Größe eines Planes auf 16 KByte reduziert. Den Aufbau der Planes können Sie aus Tabelle 3 ersehen.

Die Form der Speicheraufteilung im Grafikbereich unterscheidet sich deutlich von dem der CGA-Karte. Dieses Thema wird im VGA-Artikel dieser Grafikkarten-Serie näher erläutert.

Der CRTIC (»Cathode Ray Tube Controller«) der EGA-Karte hat wesentlich mehr Funktionen als der 6845-Baustein der früheren Bildschirmkarten. Auf die Registerbelegung und Aufgaben des neuen Video-Controllers wird ebenfalls im folgenden Artikel eingegangen.

(Markus Zietlow/sw)

VGA – das höchste der Grafikgefühle

Mit der EGA-Karte deckte IBM den Großteil der Nachfrage nach einer hochauflösenden Grafik mit guter Bildqualität ab. Für professionelle Anwender und die neue Modellreihe PS/2 entwickelte IBM die Grafikkarte VGA. Farbvielfalt und Darstellungsqualität nahmen weiter zu. Der von der EGA-Karte her bekannte 16-Farb-Modus wurde auf 640x480 Punkte ausgedehnt und die mittlere CGA-Auflösung kann nun mit 256 Farben gleichzeitig angezeigt werden. Für jede Farbe existiert die Auswahl aus einer Palette von 262144 Farben.

Im professionellen Bereich hatte sich der EGA-Standard bereits fest etabliert, und im Heimbereich stand sein Siegeszug kurz bevor, als IBM zusammen mit den PS/2-Modellen den Grafikstandard VGA einführt.

Die Bezeichnung VGA (»Video Graphics Array«) bezieht sich hauptsächlich auf einen Teil der Schaltung des Video-Systems in den PS/2-Modellen 50, 60 und 80. Das Video-Graphics-Array ist ein einziger Chip, der dieselben Funktionen beherrscht, die auf der EGA-Karte von mehreren Chips durchgeführt werden. Trotz

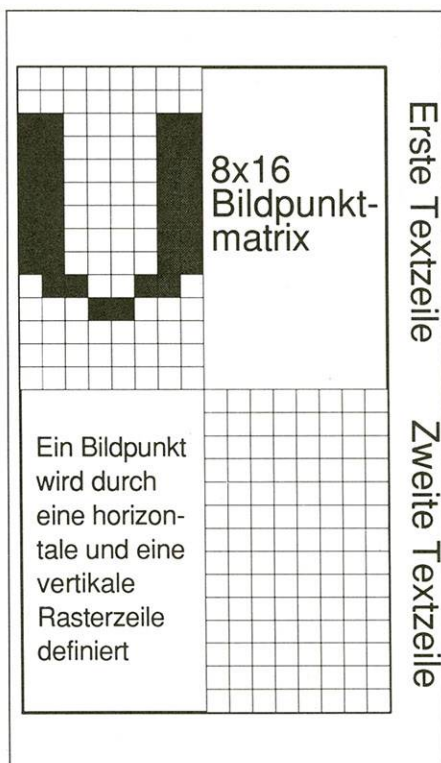


Bild 2. Matrizen bestimmen die Bildschirmdarstellung der VGA-Karte

dieser etwas irreführenden Bezeichnung hat sich der Begriff »VGA« als Bezeichnung für das gesamte Video-System etabliert.

Da die VGA-Karte zum Monochrom-Adapter sowie zu der CGA- und EGA-Karte kompatibel ist, laufen alle »sauber« konzipierten Programme problemlos. Zusätzlich besitzt die VGA-Karte einen Modus mit höherer Bildschirmauflösung (720x400 Punkte im Text- und 640x480 Punkte im Grafik-Modus) und eine erweiterte CGA mit 320x200 Punkten in 256 Farben.

Die VGA-Karte kann wesentlich mehr, als nur die Text- und Grafik-Modi der MDA- oder EGA-Karte mit einer größeren Farbpalette zu emulieren. So stellt die VGA-Karte in allen VGA-spezifischen Text-Modi ASCII-Zeichen grundsätzlich in einer Matrix von 9x16 Bildpunkten dar. Das heißt, daß jedes Zeichen aus 144 Bildpunkten zusammengesetzt ist. Im Vergleich dazu benutzt die CGA-Farbgrafikkarte nur 64, die EGA-Karte 112 und der Monochromadapter beziehungsweise die Hercules-Gratikkarte 126 Bildpunkte pro Zeichen.

Die analoge Farbdarstellung übertrifft die bisherige digitale Methode

Der gravierende Unterschied zwischen der VGA-Karte und ihren Vorgängern besteht darin, daß die übrigen Bildschirmdaten digitale RGB(Rot-Grün-Blau)-Signale erzeugen, während die VGA-Karte mit analogen Signalen arbeitet. Der Unterschied zwischen diesen beiden Systemen ist leicht zu verstehen, wenn Sie sich den Unterschied zwischen einem Lichtschalter und einem Licht-Dimmer vorstellen. Während Ihnen beim Lichtschalter nur die Möglichkeit bleibt, entweder das Licht ein-

oder auszuschalten, können Sie beim Lichtdimmer die Lichtstärke regulieren.

Ähnlich arbeitet das Funktionsprinzip der Bildschirmdaten. Bei digitalen RGB-Signalen muß der Monitor nur erkennen, ob das Signal für eine bestimmte Farbe (Rot, Grün oder Blau) gesetzt ist oder nicht. Eine Bildschirmdatenkarte, die analoge Signale verarbeitet, übersetzt die Spannung jedes Signals in entsprechende Farbsignale mit verschiedener Intensität. Aus diesem Grund benötigen Sie für die VGA-Karte entweder einen Bildschirm mit analogem Eingang oder einen Multiscan-Monitor, der beide Bildübertragungsarten kennt.

Der Grund für die analoge Bildverarbeitung liegt in der Farbvielfalt. Der Monitor kann eine wesentlich erweiterte Palette von Farbspektren darstellen. VGA arbeitet als erste Bildschirmdatenkarte anstatt mit digitalen mit analogen Bildsignalen, die neben der hohen Auflösung eine vielfältige Farbpalette erlauben. Die VGA-Karte enthält einen »Digital to Analog Converter« (»DAC«), der das Video-System in die Lage versetzt, 256 verschiedene Farben aus einer Farbpalette von 262144 oder 2^{18} Farben gleichzeitig darzustellen (Bild 1). Mit einem analogen Monochrommonitor erhalten Sie eine Farbabstufung mit 64 Grauwerten.

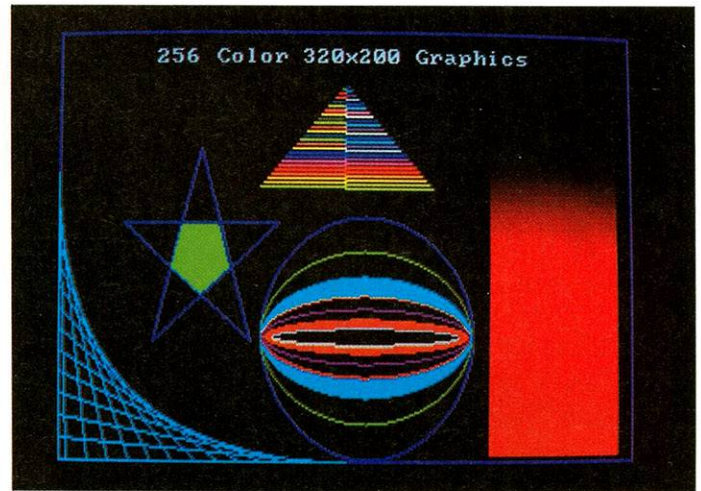


Bild 1. Der zusammen mit den PS/2-Systemen eingeführte VGA-Standard erhöhte die Farbauswahl noch einmal beträchtlich

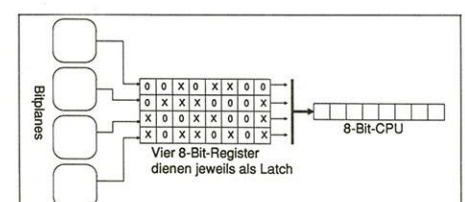


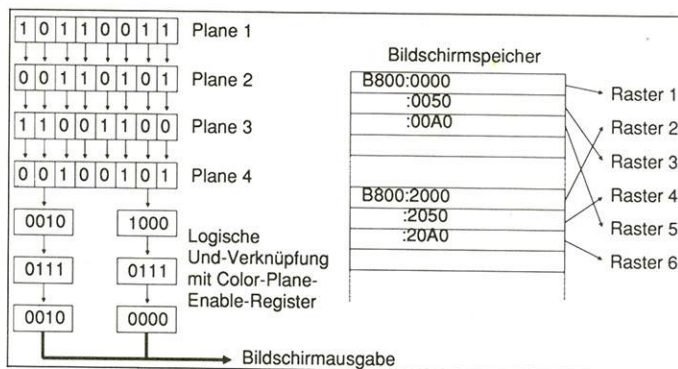
Bild 3. Das Lesen von Bildinformationen ist bei den EGA- und VGA-Karten aufwendiger als bei ihren Vorgängern

Die Bildschirmdarstellung im Text-Modus unterscheidet sich – mit Ausnahme der höheren Wiedergabequalität – in Speicheraufteilung und Wiedergabetechnik nicht von den Vorläufern der VGA-Karte. Die typische Darstellung eines Zeichens auf dem Monitor ist in Bild 2 vorgestellt. Mit dem Listing sind Sie in der Lage, ein beliebiges Zeichen direkt in den Bildschirmspeicher zu schreiben.

Der CRTC ist das Herzstück der VGA-Karte und gibt ihr auch den Namen

Das CRTC-Register der VGA-Karte entspricht in den Funktionen und dem Adressierungsbereich dem CRTC der EGA-Karte. Die Datenregister und ihre Funktionen sind in der Tabelle zusammengefasst. Ein Unterschied besteht darin, daß der VGA-Karte mehr Bitfelder zugeordnet wurden, so daß der CRTC 400- und 480-Zeilen-Raster bearbeiten kann. Zusätzlich sind – ein großer Fortschritt – alle Datenregister sowohl lesbar als auch beschreibbar. Außerdem unterstützt die VGA-Karte im

Bild 4.
Die Zeichnung verdeutlicht die Bearbeitung der Grafikinformati-
onen bis zur Bild-
schirmausgabe



Gegensatz zu ihren Vorgängern keinen Lichtgriffel. Trotz dieser Differenzen ist die Bildschirmkarte so konzipiert, daß Programme, die für die EGA-Karte geschrieben wurden, ohne Änderung auf der VGA-Karte laufen. Die EGA- und VGA-Karten werden in Assembler mit Hilfe der Anweisung »out dx,ax« programmiert, die Sie schon im Zusammenhang mit der Hercules-Grafikkarte kennengelernt haben.

Der Bildschirmspeicher der VGA-Karte ist 256 KByte groß und in vier Speicherebenen beziehungsweise

Grafik-Bildschirmebenen zu je 64 KByte aufgeteilt. Alle vier Grafik-Bildschirmebenen werden parallel adressiert und belegen deshalb nur 64 KByte Adreßbereich. Jeder Grafik-Bildschirmebene ist ein 8-Bit-Register zugeordnet (»Latch«), mit dem auch logische Operationen ausgeführt werden können. Die Struktur der Speicheraufteilung führt dazu, daß die CPU 32-Bit-Lese- und Schreiboperationen in einem Speicherzyklus durchführen kann.

Die schematische Darstellung in Bild 3 klärt Sie darüber auf, wie die CPU

C-TOOLS

C-terp der professionelle C-Interpreter.....	DM 684.-
DeSmet C Development Package inkl. Debugger jetzt V3.1 !	DM 295.-
GraphicC wiss. Präsentationsgrafik Version 4.1	DM 855.-
C_GRAPH Profi-System V2.1 Geräteunabhängige Grafik nach GKS	DM 798.-
MKS-Toolkit Unix-Tools für MS-DOS inkl. vi, awk, Kornshell etc.	DM 495.-
CDOC Dokumentationssystem für C-Programme	DM 285.-
Vance C-lib Window Bibliothek (UNIX "curses" kompatibel)	DM 295.-
MID Treiber für beliebig viele V24 - Schnittstellen	DM 285.-
PC - lint C Syntax- und Semantikprüfung über mehrere Files	DM 399.-
BTREE + ISAM File Management Routines V 2.6	DM 350.-

Zortech C++
enthält Zortech ANSI-C-Compiler
Greenleaf Super Functions nur DM 684.-

Viele Produkte mehr: Info anfordern! Keine zusätzlichen Versandkosten!

KESSLER Softwareentwicklung Mitteldorferstr. 17 3400 Göttingen Tel. 0551-792488



Interface zum Digitalisieren von Videobildern (Rekorder, TV, oder Kamera) für IBM XT/AT

VIDEO-1000 I 495,-DM

Auflösung 640 x 200 und 640 x 288 Pixel, max. 32 Graustufen. 16 KByte RAM. Unterstützt werden: CGA, HERCULES und EGA

VIDEO-1000 II 795,-DM

Auflösung 768 x 612 und 1280 x 612 Pixel, max. 32 Graustufen, 96 KByte RAM. Unterstützt werden: CGA, HERCULES, EGA und VGA.

Utilities 1 98,-DM

Diskette mit ca. 30 Programmen. Kurzfilm, 3-D Bilder, residente Programme, Laderoutinen etc. Für VIDEO-1000 I oder II.

Utilities 2 98,-DM

Digitalisieren mit verschiedenen Rastern im PAINTBRUSH-Format (Pagemaker + Ventura) max. 1536 x 1152 Pixel. Nur für VIDEO-1000 II.

Farbutilities EGA 98,-DM

Diskette incl. 3 Farbfilterscheiben zum Digitalisieren von Farbbildern mit 16 aus 64 oder 512 Farben (gerastert). Für VIDEO-1000 I und II.

Farbutilities VGA 98,-DM

Diskette incl. 3 Filterscheiben zum Digitalisieren von Farbbildern mit 16 aus 64 (640 x 480 Pixel) oder 256 Farben (320 x 200 Pixel). Für VIDEO-1000 I und II.

VIDEO-1000 VGA 1295,-DM

Echtfarb und Echtzeit-Digitizer, speziell für VGA-Grafikkarten. Auflösung 320 x 200 Pixel mit 64 Graustufen oder 256 Farben.

VIDEO-1000/256 1895,-DM

Echtzeit-Digitizer mit einer Auflösung von max. 768 x 576 Pixel und 256 Graustufen. Unterstützt werden EGA und VGA-Karten.

Info gratis. 6 Demodisketten gegen Einsendung von 30,-DM (Schein oder Briefmarken). Der Versand der Digitizer und Utilities erfolgt p.NN.

Ing.Büro Manfred Fricke

Neue Str.13, 1000 Berlin 37, Tel: 030/8015652, Telex: 181542 mfb d

(»Central Processing Unit« oder Zentraleinheit) Daten aus dem Bildschirmspeicher liest. Dieser Vorgang ist wesentlich schwieriger als beispielsweise bei der CGA- oder der Hercules-Grafikkarte. Die Zentraleinheit muß nämlich aus den vier Grafik-Bildschirmebenen die Informationen über einen Bildpunkt ermitteln.

Die Grafik in Bild 4 zeigt Ihnen ausführlich, wie ein Bildpunkt und damit ein analoges Signal für den Monitor erzeugt wird. Die Video-Einheit übernimmt die Aufgabe, aus den vier Grafik-Bildschirmebenen einen Bildpunkt für den Monitor zu erzeugen. Durch die Farbüberlappung nutzt die Einheit zur Farberzeugung sinnvoll den Bildschirmspeicher, der wie bei der CGA-Karte aufgebaut ist.

VGA – Zur Zeit die Karte mit dem besten Verhältnis von Preis und Leistung

Wer beruflich oder privat mehrere Stunden mit dem PC arbeitet und auf Farben nicht verzichten kann, für den sind VGA-Karte und Analog-Monitor fast schon ein Muß. Die VGA-Karte ist aufgrund ihrer deutlich höheren Qualität selbst gegenüber der EGA-Karte sowie insbesondere gegenüber der CGA-Karte augenscheinender und läßt dadurch bei der Arbeit am Computer eine längere Konzentrationsphase zu. Doch auch ausgesprochene Grafik-Fans werden ihre helle Freude an der VGA-Grafik haben, sofern sie gewillt sind, den dafür verlangten Preis zu bezahlen. Da die VGA-Grafik mit Sicherheit der Standard der nächsten Jahre sein wird, ist der Kauf einer VGA-Karte generell keine Fehlinvestition. Allerdings sollten alle PC-Besitzer, die es nicht so eilig haben, noch eine Weile warten. Zum einen wird in den nächsten Monaten der Preis der Karten durch die zunehmende Modellvielfalt

Der Aufbau des CRTC		
Register	Funktion	Zugriffsart
00hex	Zahl der teilweise unsichtbaren Textzeilen pro Textzeile – 1	Schreiben
01hex	Anzahl der horizontal angezeigten Rasterzeilen	Schreiben
02hex	Anfang der horizontalen Dunkelphase	Schreiben
03hex	Ende der horizontalen Dunkelphase	Schreiben
04hex	Anfang des horizontalen Zeilenrücksprungs	Schreiben
05hex	Ende des horizontalen Zeilenrücksprungs	Schreiben
06hex	Zahl der teilweise unsichtbaren Textzeilen – 1	Schreiben
07hex	Overflow-Register	Schreiben
08hex	Vordefinierte Reihe des Bildschirmrücksprungs	Schreiben
09hex	Höhe eines Zeichens in Rasterzeilen – 1	Schreiben
0Ahex	Erste Rasterzeile des Cursors	Schreiben
0Bhex	Letzte Rasterzeile des Cursors	Schreiben
0Chex	Startadresse des Bildschirmspeichers (High-Byte)	Schreiben/Lesen
0Dhex	Startadresse des Bildschirmspeichers (Low-Byte)	Schreiben/Lesen
0Ehex	Cursor-Position (High-Byte)	Schreiben/Lesen
0Fhex	Cursor-Position (Low-Byte)	Schreiben/Lesen
10hex	Start des Vertical-Retrace	Schreiben
11hex	Adresse beim Lichtgriffelimpuls (High-Byte)	Lesen
	Adresse beim Lichtgriffelimpuls (Low-Byte)	Lesen
12hex	Anzahl der vertikal angezeigten Rasterzeilen	Schreiben
13hex	Offset für die logische Verschiebung	Schreiben
14hex	Position (Unterstreichen)	Schreiben
15hex	Anfang der vertikalen Dunkelphase	Schreiben
16hex	Ende der vertikalen Dunkelphase	Schreiben
17hex	Mode Control	Schreiben
18hex	Rasterzeilen-Vergleich	

Tabelle. Die EGA- und VGA-Karten verfügen über eine große Anzahl von Datenregistern

falt deutlich fallen. Zum anderen ist dann gesichert, daß die Grafikkarten, die sich heute auf dem Markt befinden, tatsächlich 100prozentig kompatibel zum VGA-Standard von IBM sind – und darüber hinaus noch weitere Grafikmodi bieten werden, wie zum Beispiel den Modus von 640x480 Punkten in 256 Farben. Dieser Modus, der von einigen VGA-Karten bereits geboten wird, läßt die wahren Fähigkeiten der Computergrafik am PC erst richtig zur Geltung

kommen. Leider gibt es für den Aufbau dieser Bilder noch keinen einheitlichen Standard.

Doch auch IBM ist nicht untätig geblieben: Es gibt bereits die sogenannte 8515/A-Grafikerweiterung, die auf den Platinenstecker (»feature Connector«) der VGA-Karte aufgesteckt wird. Mit 8515/A wird der PC zum Grafikterminal: 1024x768 Bildpunkte in 256 Farben.

(Markus Zietlow/sw)

```
[M3A] 1: DECLARE SUB schirm.poke (reihe%, spalte%, z
      eichen%, attribut%)
[E20] 2:
[4M5] 3: ' Beispielprogramm in Quick-Basic,
[2N5] 4: ' das ein ASCII-Zeichen direkt in den
[T25] 5: ' Bildschirmspeicher 'poke't
[E20] 6:
[M76] 7: ' Veröffentlicht in der DOS Extra 5
[E20] 8:
[G94] 9: ' Autor: Markus Zietlow
[PE6] 10: ' »zeichen%« ist der ASCII-Wert des
[T65] 11: ' Zeichens, das an der Position
[8C6] 12: ' »reihe%, spalte%« in den Bildschirm
[B23] 13: ' gepokt wird
[000] 14:
[HG1] 15: schleife:
[7P6] 16: INPUT reihe%, spalte%, zeichen%, attribu
      t%
[8V8] 17: CALL schirm.poke(reihe%, spalte%, zeiche
      n%, attribut%)
[282] 18: GOTO schleife
[4M0] 19: END
[000] 20:
```

```
[PV9] 21: SUB schirm.poke (reihe%, spalte%, zeichen%,
      attribut%) STATIC
[336] 22: IF reihe% < 1 OR reihe% > 25 THEN EXIT S
      UB
[PH6] 23: IF spalte% < 1 OR spalte% > 80 THEN EXIT
      SUB
[TN6] 24: IF zeichen < 0 OR zeichen: > 255 THEN EX
      IT SUB
[UP1] 25: DEF SEG = 0
[VE4] 26: IF PEEK(&H410) AND &H40 = 48 THEN
[2C5] 27: DEF SEG = &HB000 ' Monochrom-Karte
[2R0] 28: ELSE
[1L5] 29: DEF SEG = &HB000 ' VGA, EGA & CGA-Ka
      rte
[801] 30: END IF
[626] 31: POKE (reihe% - 1) * 160 + (spalte% - 1)
      * 2, zeichen%
[5L7] 32: POKE (reihe% - 1) * 160 + (spalte% - 1)
      * 2 + 1, attribut%
[091] 33: DEF SEG
[HA1] 34: END SUB
```

Listing. Mit diesem Quick Basic-Listing können Sie jedes ASCII-Zeichen direkt in den Bildschirmspeicher schreiben

Tips und Tricks

In dieser Rubrik finden Sie alles, was Sie über das Arbeiten mit den verschiedenen Programmiersprachen wissen müssen, um Ihren PC optimal einzusetzen. Ob Pascal, C, Basic oder Assembler – zu jeder Sprache gibt es viele interessante Hinweise und Tips.

Korrektes Datum in Briefköpfen mit Turbo Pascal

Viele Textverarbeitungsprogramme erleichtern die Arbeit durch den Einsatz von Kürzeldateien. In diese Dateien legt man häufig verwendete, stets gleichbleibende Textbausteine ab. Durch Kurzformeln werden sie bei Bedarf in den zu bearbeitenden Text eingeblenet. Das fängt bereits beim Kopf von Geschäftsbriefen an: Ein Aufruf des Kürzels oder Makros »# Kopf« erzeugt beispielsweise den formatierten Briefkopf mit Namen, Adresse und Telefonnummer. Allerdings enthält der Briefkopf auch ein variables Element – das Datum. Vor allem preisgünstige Textverarbeitungsprogramme erfordern dann manuelle Korrekturen. Ein kleines Turbo-Pascal-Programm macht nun Nägel (beziehungsweise Briefe) mit Köpfen. Es sorgt dafür, daß mit dem Aufruf über die Batch-Datei das korrekte Datum von der Systemuhr in die Kürzeldatei übernommen wird (Listing 1).

Im Hauptprogramm wird das durch die Prozedur »getdate« über den MS-DOS-Interrupt 21, Funktion 42, gewonnene Systemdatum an die Variablen »d«, »m« und »j« übergeben. Für den Interrupt 21 verwendet Turbo Pascal die Standardfunktion »msdos()«. Die Prozedur »austausch« sucht in der Kürzeldatei für Geschäftsbriefe nach

dem Briefkopf und fügt dort das aktuelle Datum ein. Zur Erläuterung der Prozedur:

Der Befehl »assign« weist der Kürzeldatei »demo.dsk« für die Dauer der Prozedur die logische Datei »tdemo« zu. Mit »reset« wird der Dateizeiger auf den Anfang der Datei gesetzt. Die erste Schleife sucht nach dem Kürzelpräfix »#«, die zweite nach dem Schlüsselwort »Kopf«. Ist der Eintrag für »Kopf« gefunden, wird an der vom Briefformat vorgesehenen Stelle (hier das 150te Zeichen des Eintrags) das Datum eingetragen. Die etwas grundschulmäßige Übergabe der einzelnen Ziffern der zweistelligen Datumsangaben rührt daher, daß die logische Datei »tdemo« nur ein Zeichen lang ist und sowohl zum Suchen der Einfügestelle als auch zum Eintrag der in einzelne Zeichen umgewandelten Ziffern verwendet wird. Wenn man den Briefkopf an der ersten Stelle der Kürzeldatei ablegt (dies stellt sicher, daß sich der Eintrag durch Erweiterungen der Datei nicht ändert), so darf das umständliche Suchverfahren entfallen. Das Datum wird dann mit der Standardprozedur »seek(Dateiname,n)« eingefügt, die den Dateizeiger zur Stelle »n« bewegt.

(Werner Bogula/zi)

Pull-Down-Menüs unter Turbo Pascal 4.0

Bereits ein kurzes Programm (Listing 2) reicht aus, um nach entsprechender Erweiterung eigene Anwendungen mit einer professionellen Benutzeroberfläche zu versehen. Die Konstanten »hoch« und »breit« sind dabei entsprechend den eigenen Erfordernissen veränderbar, jedoch müssen dann die Array-Werte von »info«, »hinter« und »vorder« angepaßt werden. Bei einem Monochrom-Monitor ist die Variable »screen« mit »absolute \$B000:\$0000« zu plazieren, während man bei Farbmonitoren »absolute

```
[MX2] 1: program DATESET;
[HX6] 2: {(C) DOS International; Autor: W. Bogula}
[000] 3:
[2T3] 4: type register = record
[MH3] 5:     case boolean of true:
[FE3] 6:         (ax,bx,cx,dx,bp,si,di,
[GK3] 7:         ds,es,flags: integer);
[A01] 8:     false:
[KX2] 9:         (al,ah,bl,bh,cl,ch,
[BU1] 10:        dl,dh: byte)
[ZN0] 11:     end;
[AU2] 12:     datatype = record
[YB2] 13:         jahr : 1980..2099;
[TU1] 14:         monat: 1..12;
[9H1] 15:         tag : 1..31
[ZN0] 16:     end;
[000] 17:
[YA3] 18: var regs: register;
[LP2] 19:     datum: datatype;
[4E2] 20:     j,d,m: integer;
[000] 21:
[WN6] 22: procedure getdate(var datum: datatype);
[MU0] 23: begin
[MV1] 24:     regs.ah:=42;
[302] 25:     msdos(regs);
[663] 26:     with datum do begin
[GK2] 27:         jahr :=regs.cx;
[1P2] 28:         monat:=regs.dh;
[092] 29:         tag :=regs.dl
[4M0] 30:     end
[ZN0] 31: end;
[000] 32:
[TG3] 33: procedure austausch;
[Y92] 34: var a,z: char;
[R05] 35:     s : packed array[1..4] of char;
[4X1] 36:     t : integer;
[U43] 37:     tdemo: file of char;
[MU0] 38: begin
[XX3] 39:     assign(tdemo,'DEMO.DSK');
[482] 40:     reset(tdemo);
[P31] 41:     repeat
[000] 42:
[P31] 43:     repeat
```

```
[612] 44:         read(tdemo,a)
[BX1] 45:         until (a='#');
[000] 46:
[SF3] 47:         for t:=1 to 4 do begin
[R62] 48:             read(tdemo,a);
[LC1] 49:             s[t]:=a
[4M0] 50:         end
[000] 51:
[FH2] 52:         until (s='Kopf');
[000] 53:
[W25] 54:         for t:=1 to 149 do read(tdemo,a);
[000] 55:
[U11] 56:         z:= '.';
[BY2] 57:         a:=chr(d div 10 + 48);
[ML2] 58:         write(tdemo,a);
[CZ2] 59:         a:=chr(d mod 10 + 48);
[ML2] 60:         write(tdemo,a);
[6N2] 61:         write(tdemo,z);
[MY2] 62:         a:=chr(m div 10 + 48);
[ML2] 63:         write(tdemo,a);
[NZ2] 64:         a:=chr(m mod 10 + 48);
[ML2] 65:         write(tdemo,a);
[6N2] 66:         write(tdemo,z);
[HY2] 67:         a:=chr(j div 10 + 48);
[ML2] 68:         write(tdemo,a);
[KZ2] 69:         a:=chr(j mod 10 + 48);
[ML2] 70:         write(tdemo,a);
[000] 71:
[V62] 72:         close(tdemo);
[LZ3] 73:         write('Date set to file')
[ZN0] 74:     end;
[000] 75:
[MU0] 76: begin
[WM2] 77:     getdate(datum);
[652] 78:     d:=datum.jahr;
[PP2] 79:     m:=datum.monat;
[A33] 80:     j:=datum.jahr - 1900;
[1P1] 81:     austausch
[KN0] 82: end.
```

Listing 1. Mit »dateset.pas« erhalten Ihre Briefköpfe das aktuelle Datum

\$B800:\$0000« angibt. Der Window-Befehl nach »{fenster auf}« muß ebenfalls den Bedürfnissen angeglichen werden. Da das Programm in Turbo Pascal 4.0 geschrieben wurde, erfolgt das Öffnen und Schließen des Fensters sehr schnell.

(Achim Zeeck/zi)

```

[MX2] 1: program FENSTER;
[YS6] 2: { (C) DOS International; Autor: A. Zeeck }
[000] 3:
[5H1] 4: uses CRT;
[000] 5:
[RD2] 6: const hoch = 10;
[LP1] 7:      breit = 20;
[000] 8:
[0C5] 9: var screen : array[1..25,1..80] of
[Y11] 10:      record
[UB2] 11:      zeichen: char;
[5Y1] 12:      farbe : byte
[4M0] 13:      end
[D23] 14:      absolute $B800:$0000;
[066] 15: { bei Monochrom-Monitoren: $B000:$0000 }
[0A5] 16: info : array[1..10,1..20] of char;
[791] 17:      hinter,
[8M5] 18: vorder : array[1..10,1..20] of byte;
[732] 19: i,x,y : byte;
[R51] 20: ch : char;
[000] 21:
[EB6] 22: procedure color(vorder,hinter: byte);
[MU0] 23: begin
[TS3] 24:     textcolor(vorder);
[YU3] 25:     textbackground(hinter)
[ZN0] 26: end;
[000] 27:
[MU0] 28: begin
[1T1] 29:     randomize;
[XP1] 30:     color(7,0);
[DB1] 31:     clrscr;
[2N3] 32:     for y:=1 to 20 do begin
[YE4] 33:         color(random(7),random(7)+8);
[0K2] 34:         for i:=1 to 80 do
[784] 35:             write(chr(random(199)+32+y))
[ZN0] 36:         end;
[0X1] 37:     { retten }
[R83] 38:     for y:=1 to hoch do
[R84] 39:         for x:=1 to breit do begin
[DH5] 40:             info[y,x] :=screen[y,x].zeichen;
[MA6] 41:             hinter[y,x]:=screen[y,x].farbe shr 4;
[CN5] 42:             vorder[y,x]:=screen[y,x].farbe -
[CE3] 43:                 (hinter[y,x] shl 4)
[ZN0] 44:         end;
[222] 45:         ch:=readkey;
[U52] 46:         { Fenster auf }
[7R3] 47:         window(1,1,breit,hoch);
[BP1] 48:         color(0,5);
[DB1] 49:         clrscr;
[Z22] 50:         ch:=readkey;
[NK2] 51:         { Fenster zu }
[FH2] 52:         window(1,1,80,24);
[R83] 53:         for y:=1 to hoch do
[R84] 54:             for x:=1 to breit do begin
[F02] 55:                 gotoxy(x,y);
[3F5] 56:                 color(vorder[y,x],hinter[y,x]);
[BY2] 57:                 write(info[y,x])
[ZN0] 58:             end;
[702] 59:             ch:=readkey
[KN0] 60:         end.

```

Listing 2. »fenster.pas«, ein Grundstock für ansprechende Menügestaltung

Schnelle Sets in Turbo Pascal

Wer hat sich als fortgeschrittener Pascal-Programmierer nicht schon in der Situation befunden, innerhalb eines Programms zu überprüfen, ob ein x-beliebiger Wert in einer Wertemenge vorhanden ist? Typische Beispiele hierfür sind Filterprozesse. Turbo Pascal stellt dazu den Datentyp »set« zur Verfügung, so daß sich zum Beispiel die Abfrage, ob der Inhalt der Zeichenvariable »C« in der Menge der Großbuchstaben enthalten ist, einfach gestalten läßt:

```
if C in ['A'..'Z'] then
```

Diese Lösung läßt leider bezüglich ihres Laufzeitverhaltens stark zu wünschen übrig, weshalb schon in der Ausgabe 12'87 (Seite 62) der DOS International »gemahnt«

wurde, obiges Beispiel durch
if (C >= 'A') and (C <= 'Z') then

zu ersetzen.

So weit, so gut. Was aber tun mit

```
const
Trennzeichen : set of char =
[' ', '!', '!', '!', '(', '-', '*', '+',
',', '!', '-', '.', '!', '/', ':', '!', ';', '!', '<',
',', '=', '>', '[', ']', '{', '}'];
if C in Trennzeichen then
```

Offensichtlich hilft hier auch kein Umschreiben in eine If-Then-Anweisung mehr. Man muß also den eingebauten Test auf Inhalt durch eine eigene Funktion ersetzen. Bevor dies geschieht, sollte man sich die interne Struktur von Mengen vor Augen führen:

Mengen dürfen in Turbo Pascal maximal 256 Elemente enthalten. Jedes Element wird durch ein Bit repräsentiert, das als Kennzeichen auf Vorhandensein fungiert. Ist ein Bit gesetzt (1), so bedeutet dies, daß das Element in der Menge enthalten ist. Woher weiß man aber, welches der 256 Bit man überprüfen muß?

256 Bit belegen genau 32 Byte an Speicherplatz; die Standardlänge für Sets in Turbo Pascal. Zuerst ist nun festzustellen, in welchem Byte gesucht wird. Da ein Byte acht Bit enthält, ergibt sich diese Nummer zu

$$\text{ByteNr} := W \text{ div } 8$$

wobei »w« im vorliegenden Fall den ASCII-Wert des Zeichens und »div 8« eine ganzzahlige Division durch acht darstellt. Das Ergebnis gibt an, welches Byte (0 bis 31) überprüft werden muß. Der bei der Division entstandene Rest

$$\text{BitPos} := W \bmod 8$$

gibt die Bit-Nummer (0 bis 7) innerhalb dieses Bytes an.
Im Dualsystem lassen sich nun »W div 8« durch »W shr 3«
und »W mod 8« durch »W and 7« ersetzen.

Um eine Maske »m« zu erhalten, führen Sie die Operation

```
M := 1 shl BitPos
```

durch. Erfährt diese Maske mit dem Byte des Sets, das durch »bytenr« festgelegt ist, eine logische Und-Verknüpfung und hat ferner das gewonnene Ergebnis einen Wert ungleich Null, so ist »c« in der Menge enthalten.

Die Funktion »contains« (Listing 3) testet einen Wert »i« auf Vorhandensein in einer Menge »s« irgendeines zugrundeliegenden Typs. Sie ist unter Turbo Pascal 4.0 doppelt so schnell wie die eingebaute Funktion (Listing 4).

(Jörg Meinertz/zi)

```

[U27]      1: function Contains(I:byte; var S): boolean;
[MU0]      2: begin
[Y4]       3:     inline($4C/           { DEC SP           }
[BX4]      4:     $C4/$BE/$/           { LES DI,S[BP]    }
[XT4]      5:     $8A/$86/I/           { MOV AL,I[BP]    }
[N84]      6:     $32/$E4/           { XOR AH,AH       }
[F04]      7:     $B3/$01/           { MOV BL,1        }
[2B4]      8:     $8A/$C8/           { MOV CL,AL       }
[934]      9:     $D0/$E8/           { SHR AL,1        }
[934]     10:    $D0/$E8/           { SHR AL,1        }
[934]     11:    $D0/$E8/           { SHR AL,1        }
[D74]     12:    $03/$F8/           { ADD DI,AX       }
[844]     13:    $80/$E1/$07/       { AND CL,7        }
[V94]     14:    $D2/$E3/           { SHL BL,CL       }
[TV4]     15:    $26/$84/$1D/        { AND BL,ES:[DI]  }
[T64]     16:    $75/$04/           { JNZ TRUE        }
[L84]     17:    $32/$C0/           { XOR AL,AL       }
[Y14]     18:    $EB/$02/           { JMP SHORT EXIT  }
[015]     19:    $B0/$01/           { TRUE: MOV AL,1   }
[CX5]     20:    $8B/$46/$FF); { EXIT: MOV [BP-1],AL }
[ZN0]     21: end;

```

Listing 3. »contains.inc«, die Funktion ersetzt den In-Operator in Turbo Pascal


```
[2B2] 1: program SETS;
[DH5] 2: { Das Programm demonstriert den
[FX4] 3: Laufzeitunterschied }
[000] 4:
[VD4] 5: var TestSet : set of byte;
[733] 6: Counter : integer;
[RW2] 7: Result : boolean;
[000] 8:
[T13] 9: {$I CONTAINS.INC}
[000] 10:
[MU0] 11: begin
[US3] 12: TestSet:=[1..10,22..25,50,
[K12] 13: 60..86,100..200];
[YE4] 14: if (Contains(5,TestSet)) and
[X25] 15: (not Contains(55,TestSet)) then
[ZL5] 16: writeln('Contains funktioniert!')
[134] 17: else writeln('Contains '+'
[NR3] 18: ' funktioniert nicht!');
[4R6] 19: writeln('Eingebaute Pascal-Funktion: ');
[991] 20: readln;
[E93] 21: write('G'); { Beep }
[F04] 22: for Counter:=1 to 30000 do
[3H3] 23: Result:=1 in TestSet;
[0R1] 24: write('G');
[6L3] 25: writeln('Contains: ');
[991] 26: readln;
[0R1] 27: write('G');
[F04] 28: for Counter:=1 to 30000 do
[6T4] 29: Result:=Contains(1,TestSet);
[FH1] 30: write('G')
[KN0] 31: end.
```

Listing 4. »sets.pas« demonstriert den Laufzeitenunterschied

Kontrollierter Programmabbruch unter Turbo Basic

Besonders in Basica geschriebene Programme der Public-Domain-Szene lassen einen Programmabbruch häufig nur mit der Tastenkombination [Ctrl-Break] zu. Diejenigen Anwender, die Programme dieser Art mit Turbo Basic compilieren, sind dann vom Ergebnis im wahrsten Sinne des Wortes gefesselt: Ein »Entrinnen« – zurück ins Betriebssystem – gibt es nicht. Nach dem Start eines solchen Programms hilft nur noch der oft geübte »Spreizgriff« [Ctrl-Alt-Del].

Um die Kombination [Ctrl-Break] während der Programmausführung zu nutzen, müssen Sie im Options-Menü der Turbo-Basic-Benutzeroberfläche den Schalter »Keyboard Break« auf »on« setzen, bevor Sie Ihr Programm compilieren beziehungsweise ausführen. Beachten Sie bitte, daß [Ctrl-Break] Ihr Programm nur dann unterbricht, wenn der Schalter »Keyboard Break« auf »on« gesetzt ist und das Programm über den Bildschirm oder einen Drucker Ein- und Ausgaben ausführt. Ist im Debug-Menü der Schalter »Trace« auf »on« gesetzt, so läßt sich ein Programm ebenfalls mit [Ctrl-Break] unterbrechen.

(zi)

Mehr Stellen – mehr Platz

Wenn Sie den Befehl »print using« zur Darstellung einfach- oder doppeltgenauer Realzahlen anwenden, kann es vorkommen, daß Turbo Basic zusätzliche Stellen liefert und damit den vorgegebenen Rahmen überschreitet. Der Grund dafür ist im internen Zahlenformat des Compilers zu finden. Turbo Basic rechnet intern mit einem sehr präzisen 80-Bit-Format. In Basica hingegen werden Zahlen nur mit 24 Bit verarbeitet. Daher entstehen bei Turbo Basic unter Umständen zusätzliche Dezimalstellen, die unter Basica bei einer Rundung einfach entfallen. Um diesem Umstand Rechnung zu tragen, sollten Sie beim Arbeiten mit »print using« unter Turbo Basic etwas mehr Platz bei der Zahlenausgabe vorsehen oder existierende Programme entsprechend ändern.

(zi)

Rätselhaftes Ende einer Zahl

Einige Versionen von GW-Basic legen ein etwas seltsam anmutendes Verhalten an den Tag. Versucht man eine Zahl formatiert auszugeben (mit »print using«) und stellt man im Formatstring nicht genügend Platz für die Darstellung eines Minuszeichens zur Verfügung, so entfernt GW-Basic automatisch die letzte Ziffer der Zahl. Um die Zahl nicht zu »verstümmeln«, sollte man deshalb (innerhalb des Formatstrings) mehr Platz für die Ausgabe von Zahlen vorsehen. Turbo Basic und Basica verhalten sich »disziplinierter« – sie lassen Dezimalstellen nicht einfach verschwinden. Wenn Sie in diesen Basic-Varianten ein Minuszeichen bei der formatierten Zahl ausgabe darstellen wollen, müssen Sie es im Print-Using-Befehl berücksichtigen.

(zi)

Schnelle Spiele ganz langsam

Je höher die Taktgeschwindigkeit bei einem Computer ist, desto schneller werden die Programme abgearbeitet, und lästige Wartezeiten verringern sich. Auch die Art des Prozessors ist für die Geschwindigkeit verantwortlich. In der Regel wird ein Geschwindigkeitszuwachs als positiv empfunden. Doch bei Spielen ist dieser Effekt nicht immer wünschenswert, da durch die hohe Ausführungsgeschwindigkeit die Reaktion des Spielers manchmal überfordert und somit das Programm nicht spielbar wird. Das Turbo-C-Programm »bremse« (Listing 5) verhindert unerwünscht hohe Spielgeschwindigkeiten.

Die Wirkungsweise läßt sich wie folgt erklären. Der im Computer eingebaute Timerbaustein ruft die Routine, auf die der achte Zeiger der Interrupt-Vektortabelle zeigt, ungefähr 18,2 mal pro Sekunde auf. Normalerweise ist das der Systemtakt. Das Programm leitet den Interrupt-Vektor jedoch auf eine Routine um, die eine Warteschleife ausführt und erst danach den Timer aufruft.

Das Programm erwartet als Parameter einen numerischen Wert. Dieser Wert gibt an, wie oft die Warteschleife durchlaufen wird. Daraus ergibt sich, daß die Ausführungsgeschwindigkeit mit der Höhe des Wertes abnimmt. Man muß darauf achten, daß der übergebene Wert nicht zu groß wird, denn wenn die Warteschleife länger als 1/18,2 Sekunden zur Ausführung braucht, wird der Interrupt 8 wieder aufgerufen, bevor die Schleife zu Ende war. Das Resultat ist eine wunderschöne Endlosschleife, mit der Folge, daß das ganze System »hängt«.

Auf einem 8-MHz-PC mit V30-Prozessor sind Werte bis zirka 3000 zulässig. Ein 12-MHz-286-AT verkraftet Werte bis zirka 50000.

Mehrfachaufrufe von »bremse« sind nicht zulässig. Vor einem Aufruf mit einem neuen Wert muß »bremse« durch die Eingabe von »exit« beendet werden.

(Ralf Schliepat/tr)

```
[W60] 1: /*
[344] 2: BREMSE von Ralf Schliepat
[170] 3: */
[000] 4:
[9L2] 5: #include <dos.h>
[L13] 6: #include <stdio.h>
[U83] 7: #include <stdlib.h>
[000] 8:
[RP4] 9: void interrupt clockticks();
[000] 10:
[H72] 11: int ticks = 0;
[000] 12:
[0N4] 13: main(int argc, char *argv[])
[P70] 14: {
[KY6] 15: if (argc != 2 || (ticks = atoi(argv[1])) =
= 0)
[P70] 16: {
[X8C] 17: printf("Falsche Parameterangabe\007\nAufruf: BREMSE (Verzögerung)\n");
```



```
[UL1] 18: return(1);
[U70] 19: }
[ZV1] 20: if (ticks > 0)
[P70] 21: {
[H0B] 22: printf("BREMSE, Version 1.3, Copyright (c
) 8/1988, Ralf Schliepat\n");
[2VA] 23: printf("\nVerzögerung: %d Ticks\nZurück m
it EXIT\n", ticks);
[YK9] 24: setvect(0x67, getvect(8));
/* Sichere Timer auf 0x67 */
[MGA] 25: setvect(8, clockticks); /* clockticks
auf alten Timer (8) legen */
[T03] 26: system("COMMAND");
[A1B] 27: setvect(8, getvect(0x67)); /* Timer
wieder auf 8 zurücksetzen */
[U70] 28: }
[U70] 29: }
[000] 30: }
[WM4] 31: void interrupt clockticks()
[P70] 32: {
[F11] 33: int i;
[000] 34: }
[WS3] 35: for(i = 0; i < ticks; i++) ;
[169] 36: geninterrupt(0x67);
/* Löse Timer aus */
[U70] 37: }
```

Listing 5. »bremse.c«, eine Software-Bremse in Turbo C

Eine Turbo-Pascal-Stoppuhr

Listing 6 ist eine Unit für Turbo Pascal 4.0 und stellt Ihnen drei nützliche Funktionen zum Optimieren von Programmen zur Verfügung: »start«, »stop« und »zeit«. Die Anwendung ist ganz einfach. Am Anfang des jeweiligen Programms erfolgt die Implementierung der Unit durch

uses time;

Gestartet wird die Stoppuhr mit »start« und angehalten mit »stop«. Der aktuelle Stand der Uhr in Hundertstelsekunden läßt sich mit der Funktion »zeit« abfragen (zum Beispiel »writeln(zeit)«). Der Wert dieser Funktion wird in Form einer Longinteger-Variable gespeichert. Eine genauere Messung ist zumindest mit der normalen PC-Uhr nicht sinnvoll.

(Vojtech Terber/tr)

```
[MR1] 1: unit time;
[000] 2:
[SK1] 3: interface
[000] 4:
[VZ3] 5: { Unit benutzt DOS-Uhr }
[VG1] 6: uses dos;
[000] 7:
[CF3] 8: { Starten der Uhr }
[YW2] 9: procedure start;
[000] 10:
[RF3] 11: { Stoppen der Uhr }
[NS2] 12: procedure stop;
[000] 13:
[L76] 14: { zeit=Zeit zwischen start und stop
in 1/100 sek }
[X22] 15:
[H24] 16: function zeit: longint;
[000] 17:
[4L2] 18: implementation
[RN0] 19: var
[HB5] 20: h1,h2,m1,m2,sec1,sec2,hs1,hs2: word;
[000] 21:
[YW2] 22: procedure start;
[MU0] 23: begin
[9F7] 24: { Einlesen der Startzeit über die DOS-Uhr }
[VH3] 25: gettime(h1,m1,sec1,hs1)
[ZN0] 26: end;
[000] 27:
[NS2] 28: procedure stop;
[MU0] 29: begin
[S26] 30: { Einlesen der Endzeit über die DOS-Uhr }
[6K3] 31: gettime(h2,m2,sec2,hs2)
[ZN0] 32: end;
[000] 33:
[RG2] 34: function zeit;
[MU0] 35: begin
[Y15] 36: { Berechnen der Zeitdifferenz
und umrechnen in 1/100 Sekunden }
[4B5] 37: zeit:=(h2-h1)*360000+(m2-m1)*6000+(sec2-s
ec1)*100+hs2-hs1
[ZN0] 39: end;
[000] 40:
[RN0] 41: end.
```

Listing 6. »time.pas«, eine nützliche Turbo-Pascal 4.0-Unit zum Optimieren von Programmen

KAYSER Daten-Technik

J. Kayser Gladiolenweg 2a 4796 Salzkotten
Telefon 05258/3063 Telefax 05258/4298

Software für IBM* Personalcomputer
XT – AT – PS/2

Professionell in Qualität und Preis

Neu! Neu! Neu! Neu! Neu!

MIP Kasse	Kassenbuch und Einnahme-/Überschubrechnung für Gewerbe und Privatpersonen, täglicher Kassenstand, freie Kontenverwaltung, USt-Listen, Journal, Taschenrechner. 1 Diskette 5,25 Zoll, 1 Diskette 3,5 Zoll, Handbuch	69,-
MIP Büro	Internationales Büropaket mit Rechnungswesen, Lager, Versand, Postvertrieb, Buchhaltung, Mahnwesen, offene Posten, Präsentationsgrafik, Textverarbeitung, Termine, Stammdaten, freie Steuersätze, freie Formulargestaltung, freiwählbare Währung. 1 Diskette 5,25 Zoll, 1 Diskette 3,5 Zoll, Handbuch	299,-
MIP-Text Plus	Textverarbeitung und Programmeditor, Zeilenumbruch, Blocksatz, Makros, Suchfunktion, Serienbriefe, Adresskartei mit Notizblatt je Adresse, Sortierroutinen auf Bildschirm und Drucker. 1 Diskette 5,25 Zoll, 1 Diskette 3,5 Zoll, Handbuch,	69,-
MIP-Druck Plus	Programm zum Erstellen beliebiger Etiketten mit verschiedenen Schriftarten und Seriendruck, speicherbar, Formulardruck, Überweisungen, Scheckdruck. 1 Diskette 5,25 Zoll, 1 Diskette 3,5 Zoll, Handbuch,	69,-
MIP-Post	Dieses Programm verwaltet Ihren kompletten Postvertrieb mit Druck von Paketkarten, Nachnahme-Karten, Briefnachnahme, Paketaufkleber, Adresskartei Notizen. 1 Diskette 5,25 Zoll, 1 Diskette 3,5 Zoll, Handbuch,	69,-
MIP-Aktie	Programm zum Verwalten von Aktien und Optionscheinen in 9 verschiedenen Depots mit Depotbestand, Gewinn-/Verlustrechnung, aktuelle Kurse, Druckroutine. 1 Diskette 5,25 Zoll, 1 Diskette 3,5 Zoll, Handbuch,	69,-
MIP-Schule	Programm zum Verwalten von Schülerstammdaten mit Sortierroutine auf Bildschirm und Drucker, Klassenliste, Fahrtschülererfassung, Schullaufbahn, Schulabgang. 1 Diskette 5,25 Zoll, 1 Diskette 3,5 Zoll, Handbuch,	69,-
MIP-Maus	Jede Software läuft über dieses Programm mit der Maus (z.B. Dbase, Framework)* Die Empfindlichkeit der Maus ist frei wählbar, speicherresistent	29,-
Maus-Set	Genius GM 6 plus Maus und MIP-Maus incl. Grafikprogramm und Mauspad	120,-
MIP-Fakt	Komplettes Fakturierungsprogramm mit Kunden-/Lieferantenkartei, Rechnungen Mahnungen, Buchhaltung, Statistik mit Präsentationsgrafik, Datensicherung	29,-
MIP-Lager	Lagerverwaltungsmodul für MIP-Fakt; Artikel können direkt in die Rechnung übernommen werden; Preislisten, Inventar, Artikelaufkleber, Einlagen	29,-
MIP-Dos	Lernprogramm für die wichtigsten Dos-Befehle und deren Einsatz	29,-
MIP-Term	Terminplaner mit Adresskartei, Sortierroutinen auf Bildschirm und Drucker, Etiketten und Notizblatt je Adresse	29,-
	Diskettenbox incl. 10 Disk 3,5 Zoll 2 DD	45,-
	Diskettenbox incl. 10 Disk 5,25 Zoll 2 D	29,-
	Diskettenbox incl. 5,25 Zoll 2 HD	45,-

Händleranfragen erwünscht * eingetragene Warenzeichen

Tips & Tricks für Einsteiger

Der Käufer eines PC bekommt einen Computer, der nach allen Seiten offen für Experimente und Erweiterungen ist – genau das richtige für einen Computer-Freak. Am Anfang tut er sich verständlicherweise jeder noch etwas schwer, vor allem, wenn es um grundlegende Verständnisprobleme geht. Auf diesen Seiten finden Sie wichtige Hinweise über Diskettenlaufwerke und Festplatten, die Sie in den wenigsten Handbüchern finden werden, weil sie ja »so selbstverständlich« sind.

Es gibt bei PCs drei weit verbreitete Laufwerktypen. Sie unterscheiden sich in Diskettenformat (also der Größe der verwendeten Disketten) und Speicherkapazität: 360 KByte-5 1/4 Zoll, 1,2 MByte-5 1/4 Zoll und 720 KByte-3 1/2 Zoll. In Deutschland ist die Maßeinheit Zoll ziemlich ungebräuchlich. Die 5 1/4-Zoll-Disketten sind zirka 13x13 Zentimeter groß und leicht biegsam. Die Diskette selbst steckt in einer Papierhülle zur Aufbewahrung. 3 1/2-Zoll-Disketten haben eine sehr stabile Plastik-Ummantelung und benötigen keine separate Aufbewahrungshülle. Achten Sie beim Kauf von 3 1/2-Zoll-Disketten unbedingt darauf, daß Sie nicht versehentlich die 3-Zoll-Versionen kaufen, mit denen Sie nichts anfangen können. Sie sehen ähnlich aus wie 3 1/2-Zoll-Disketten und wurden speziell für die Computer der Schneider CPC-Reihe entwickelt.

Wenn Sie sich einen PC-XT mit einem 8088- oder 8086-Prozessor gekauft haben, verfügen Sie mit sehr großer Wahrscheinlichkeit auch über Standardlaufwerke, die eine Diskette doppelseitig mit 360 KByte Kapazität formatieren können. Diese Laufwerke sind robust und problemlos. Sie müssen beim Kauf von Disketten auch nicht unbedingt auf eine bestimmte Qualität achten (Backup-Disketten ausgenommen, dazu jedoch später mehr).

Bei PC-ATs mit einem 80286- oder 80386-Prozessor sind meist 1,2-MByte-Laufwerke eingebaut. Diese unterscheiden sich äußerlich nicht von den 360-KByte-Versionen. Woran erkennt man sie dann? Sie müssen dazu eine Diskette auf dem fraglichen Laufwerk formatieren. Starten Sie Ihr System, indem Sie eine MS-DOS-System-Diskette einlegen und den Computer einschalten. Rufen Sie das Format-Programm durch die Eingabe von »format a:« auf. Danach legen Sie eine leere Diskette in Ihr Laufwerk und betätigen die Return-Taste. Wenn Ihr Computer den Formatierungsvorgang ohne Probleme durchführt und anschließend eine Speicherkapazität von rund 360 KByte meldet, verfügen Sie über Standardlaufwerke.

Wenn Sie 1,2-MByte-Laufwerke eingebaut und eine normale Diskette zum Formatieren verwendet haben, dürfte Ihr Laufwerk nun protestieren. Dies macht sich zum Beispiel durch schabende Geräusche bemerkbar. Eventuell bricht der Computer das Formatieren der Diskette sogar ganz ab und meldet »Ungültiges Medium« oder ähnliches. Was ist passiert?

Für 1,2-MByte-Laufwerke benötigen Sie besondere »High-Density-Disketten« (zumindest dann, wenn Sie die volle Kapazität nutzen möchten). Diese sind ziemlich teuer, weisen jedoch eine wesentlich feinere Beschichtung auf. Dadurch passen auf eine »Diskettenspur« mehr Bytes. Wichtig: Auf 1,2-MByte formatierte Disketten können Sie mit einem 360-KByte-Laufwerk nicht mehr einlesen, nur noch neu formatieren.

Auf Ihrem 1,2-MByte-Laufwerk lassen sich aber auch normale, preiswerte Disketten formatieren. Dann allerdings beträgt die Kapazität nur noch 360-KByte. Dem Format-

Programm wird dies durch die Option »/4« mitgeteilt. Also »format a: /4«.

Übrigens stellt sich das 1,2-MByte-Laufwerk automatisch auf das verwendete Diskettenformat ein. Es erkennt also selbständig, ob es sich bei der gerade eingelegten (und bereits formatierten) Diskette um eine High-Density- oder um eine Standard-Diskette handelt.

Ein 3 1/2-Zoll-Laufwerk hat neben seinen Vorteilen auch diverse Nachteile aufzuweisen

Noch ein paar Worte zu den 3 1/2-Zoll-Laufwerken. Diese Geräte erfreuen sich zunehmender Beliebtheit, da das verwendete Diskettenmaterial durch die stabile Ummantelung sehr robust ist. Diese Laufwerke weisen jedoch auch einige Nachteile auf, die meist verschwiegen werden. Zum einen wird der größte Teil der Software, die Sie für Ihren PC kaufen können, nur auf 5 1/4-Zoll-Disketten geliefert. Weiterhin sind nach unseren Erfahrungen diese Laufwerke alle sehr geräuschintensiv. Vor allem in Laptops (tragbare PCs) wird meist sehr wenig zur Geräuschdämpfung getan. Und schließlich sind die 3 1/2-Zoll-Disketten auch etwas teurer.

Ein 3 1/2-Zoll-Laufwerk eignet sich hervorragend als Zweitgerät neben einer 5 1/4-Zoll-Version, zum Beispiel zur Herstellung von Backups.

Eine nagelneue Diskette, frisch aus der Packung, ist physikalisch leer. Auf der magnetischen Oberfläche befinden sich noch keinerlei Strukturen. Um nun auf diese Diskette Daten und Programme speichern zu können, muß sie erst »formatiert« werden. Dabei wird die Diskette in 40 oder 80 Spuren mit meist 15 Sektoren pro Spur und 512 Bytes pro Sektor eingeteilt. Weiterhin legt das Format-Programm auf der ersten (äußersten) Spur einige Informationen über den Aufbau der Diskette ab. Diese werden zum Beispiel benötigt, wenn Sie »dir« eingeben.

Zum Verständnis des Formatierungsvorgangs ist es wichtig zu wissen, daß auch eine »leere«, aber formatierte Diskette bereits auf jeder Spur und auf jedem Sektor Daten enthält. Das hat nichts damit zu tun, ob der Dir-Befehl Ihnen nun meldet, daß die Diskette keine Programme enthält oder nicht. Für das Laufwerk ist es unwichtig, was Sie auf die Diskette schreiben. Selbst wenn ein Sektor »leer« ist, so enthält er doch lauter Nullen. Und für den Computer ist eine Null eine Zahl wie jede andere auch. Wird jetzt ein Programm auf die Diskette gespeichert, so überschreibt das Laufwerk einfach die bereits vorhandenen Nullen.

Es dauert, bevor Sie als Festplatten-Besitzer mit Platzproblemen konfrontiert werden

Festplatten sind eine feine Sache, aber was ist bei diesen Datenträgern besonders zu beachten? Sie können einmal von der Kapazität her eine Unmenge an Daten aufnehmen. Es wird in den meisten Fällen Monate dauern, bis Sie auf einer Festplatte ernsthafte Platzprobleme bekommen. Und außerdem sind »Harddisks«, wie der Profi sagt, sehr schnell.

Diese sehr hohe Datendichte hat leider auch ihren Preis (nicht nur in Mark ausgedrückt): Ein Festplattenlaufwerk ist der mit Abstand empfindlichste Teil einer Computeranlage. So kann es bereits auf geringfügige Erschütterungen mit Totalausfall reagieren. Die Schreib/Leseköpfe (meist sind es vier oder sechs) rasen mit einer sehr hohen Geschwindigkeit über die Platten, und das in einem Abstand von nur wenigen tausendstel Millimetern. Man kann sich leicht vorstellen, was geschieht, wenn der Kopf auch nur ganz kurz auf der Oberfläche aufsetzt: Im günstigsten Fall wird nur ein Teil der Platte völlig unbrauchbar (mitsamt der darauf gerade enthaltenen Daten), weil die empfindliche Beschichtung am Aufsetzpunkt zerstört

wird. Im ungünstigsten Fall ist der Schreib/Lesekopf danach nicht mehr zu gebrauchen – und damit das gesamte Laufwerk. Der Fachmann nennt den Vorgang des Aufsetzens »Head Crash«.

Was kann man als Laie aus diesen Tatsachen lernen?

1. Wenn Sie Ihren Computer (mit Festplatte) transportieren, lassen Sie vorher ein Parkprogramm laufen. Ein solches wird zu den meisten Platten gleich mitgeliefert. Es fährt die Schreib/Leseköpfe auf die innerste Plattenspur, die meistens keine Daten enthält. Bei manchen Platten hat diese Spur bereits keine magnetische Beschichtung mehr, sondern eine spezielle, kopfschonende Auflage. Falls an dieser Stelle ein Head-Crash passieren sollte, sind die Datenverluste gleich null. Die innerste Spur heißt treffenderweise »Park-Zone« oder auch »landing zone«.

2. Machen Sie regelmäßig, mindestens jedoch einmal die Woche ein Plattenbackup. Dabei werden alle Daten von der Platte auf Disketten kopiert. Wie das genau geht, erfahren Sie weiter unten.

3. Geben Sie Ihrer Festplatte die Gelegenheit, sich wechselnden Umgebungstemperaturen anzupassen. Beispiel: Der Computer lag im Winter längere Zeit im Auto und kommt dann in die warme Wohnung. Es kann nun passieren, daß sich einige Elemente der Platte minimal verziehen. Lassen Sie den Computer lieber noch ein oder zwei Stunden ausgeschaltet.

4. Sobald sich auch nur ein einziger Lesefehler auf der Platte bemerkbar macht, fertigen Sie umgehend ein komplettes Backup. Danach müssen Sie mit einem »Low-Level-Format«, einem Vor-Formatierprogramm, die Platte neu einrichten. Dabei werden defekte Bereiche als ungültig gekennzeichnet. Sollte bei Ihrer Festplatte kein solches Programm beiliegen, fragen Sie Ihren Händler. Meist wird er es Ihnen kostenlos kopieren.

Das regelmäßige Backup ist ein Muß für einen pflichtbewußten Festplatten-Benutzer

Was ist nun eigentlich ein Backup? Wie weiter oben bereits festgestellt wurde, sind Festplatten hochempfindliche Einheiten. Hinzu kommt, daß Platten meist sehr viele wichtige Daten enthalten (denken Sie nur an einen Betrieb, der seine gesamte Kundendatei auf einer Festplatte gespeichert hat).

Nun stellen Sie sich vor, daß Sie irgendein neues Programm ausprobieren, dessen genaue Funktionsweise Sie noch nicht kennen. Sie betätigen eine falsche Taste (eine einzige kann ausreichen) und – Ihre Festplatte ist leer. Solche Programme gibt es leider zur Genüge. Was tut der Profi dagegen? Er macht regelmäßig ein Backup seiner Festplatte. Das bedeutet einfach ausgedrückt, daß von der kompletten Platte eine Sicherheitskopie auf Disketten angelegt wird. Sollte die Platte aus irgendeinem Grund plötzlich leer sein, kann immer noch auf die Sicherheitskopie auf den Disketten zugegriffen werden.

Es gibt mittlerweile eine große Anzahl an Programmen, die diesen Vorgang automatisieren. Sie brauchen nur noch die Disketten zu wechseln, alles weitere erledigt die Software automatisch. Bei den neuesten MS-DOS-Versionen wird ein Backup-Programm gleich mitgeliefert. Treffenderweise heißt es »backup«. Das entsprechende Programm zum Zurückkopieren der Daten von Diskette auf Festplatte heißt »restore«.

Wann sollte ein Backup gemacht werden? Man könnte natürlich sagen »täglich« oder »wöchentlich«, was aber nicht besonders sinnvoll ist. Sie können sich so helfen: Überlegen Sie sich, wann Sie das letzte Mal ein Backup gemacht haben. Dann denken Sie nach, welche Programme und Daten seit dem letzten Backup neu auf Ihre Festplatte gekommen sind. Wenn Sie das Gefühl haben, daß Sie der Verlust eben dieser Daten sehr schmerzhaft treffen würde, dann ist es Zeit für eine neue Festplattensicherung.

Verwenden Sie bei Backups nur hochwertiges Diskettenmaterial. Investieren Sie ruhig etwas mehr Geld für Markendisketten. Bedenken Sie, daß dies ja eine einmalige Anschaffung ist.

Zwischen einer Festplatte und einer Diskette gibt es einen wesentlichen Unterschied. Wird eine Diskette formatiert, so sind danach alle Daten, die diese Diskette vorher eventuell enthielt, gelöscht. Sie werden mit Nullen überschrieben.

Wie Sie aus den vorangegangenen Beiträgen schon wissen, muß eine Festplatte ganz zu Beginn mit einem Vor-Formatierprogramm initialisiert werden. Dieser einmalige Durchlauf schreibt die Datenstruktur (also die Einteilung in Spuren und Sektoren) auf die Platte. Nur bei diesem Vorgang sind die Daten danach wirklich physikalisch von der Platte gelöscht.

Der »normale« Format-Befehl (»format c:«) überprüft lediglich, ob sich nicht noch irgendwo Lesefehler auf der Platte befinden und löscht anschließend alle Daten aus den Directory-Tabellen. Die Platte ist nachher scheinbar leer, die Programmdateien jedoch sind noch in den einzelnen Sektoren vorhanden.

Diesen Umstand machen sich einige »Re-Formatierprogramme« zunutze: Wurde versehentlich die Festplatte gelöscht, dann kann man mit Hilfe dieser Programme den alten Inhalt wieder herstellen (vorausgesetzt, man hat noch keine neuen Daten auf die Platte geschrieben). Die zwei bekanntesten Re-Formatierprogramme sind »mirror« aus den PC-Tools und »format-recover« aus den Norton-Utilities.

Und noch etwas: Schalten Sie Ihren Computer niemals aus, wenn er gerade auf die Platte zugreift. Die Wahrscheinlichkeit eines Head-Crashes ist sehr hoch.

Haben Sie versehentlich das Format-Programm auf Ihrer Festplatte gestartet, dann drücken Sie einfach [Alt-Ctrl-Del] (Reset). Ihre Daten bleiben dann erhalten. Dies ist allerdings nur so lange möglich, wie das Programm noch die einzelnen Köpfe und Cluster der Festplatte hochzählt. Sobald die Directory-Spuren einmal neu geschrieben sind, hilft, wenn überhaupt, nur noch ein Re-Format-Utility.

Dieser Trick funktioniert leider nur bei Festplattenlaufwerken. Haben Sie einen versehentlichen Formatierungsbefehl für eine Diskette ausgelöst, so gibt es leider keinen Weg mehr, um Ihre Daten noch zu retten.

Was passiert eigentlich beim Booten – oder die geheimnisvollen Boot-Disketten

Vor allem Umsteiger von anderen Computern tun sich relativ schwer, den Begriff »Boot-Diskette« zu verstehen. Ein PC hat sein Betriebssystem nicht in einem ROM-Baustein fest gespeichert, sondern lädt es beim Einschalten erst von der Diskette. Dies hat den Vorteil, daß die Software-Häuser relativ problemlos zum Beispiel neue Betriebssysteme entwickeln können.

Trotzdem enthält jeder PC schon beim Einschalten einige Programmroutinen. Diese sind im sogenannten BIOS-ROM gespeichert. Die wichtigste Routine ist diejenige, die den Computer veranlaßt, sich das Betriebssystem von Diskette zu laden. Denn auch das muß natürlich von einem Programm gesteuert werden.

Um eine Boot-Diskette zu erstellen, benötigen Sie das MS-DOS-Format-Programm. Durch die Option »/s« teilen Sie diesem Format-Programm mit, daß das Betriebssystem auf die neue Diskette übertragen werden soll (also zum Beispiel »format a: /s«). Danach ist diese Diskette »bootfähig«.

(tr)

Tips und Tricks zu MS-DOS

Sie glauben, Sie wüßten schon alles über MS-DOS? Na, dann lassen Sie sich mal von den Tips und Tricks zu MS-DOS überraschen. – Hätten Sie beispielsweise gedacht, daß man in einer Batch-Datei durch Voranstellen eines »@« die Ausgabe eines Befehls auf dem Bildschirm verhindern kann? Oder wissen Sie, wie man sich den Inhalt des Boot-Sektors ansehen kann, ohne gleich einen Diskettenmonitor kaufen zu müssen? Es gibt noch eine ganze Menge Wissenswertes rund um das Betriebssystem.

Erzeugen einer Datei mit dem Copy-Befehl

Mit dem Copy-Befehl kann man einfach und schnell kurze Dateien herstellen. Dieser Befehl ist immer verfügbar, wenn man sich auf der MS-DOS-Kommandoebene befindet. Geben Sie ein:

```
copy con <Laufwerk: \ Pfad \ Dateiname> [Return]
```

Der Parameter »con« des Copy-Befehls gibt an, daß die Tastatur als Eingabegerät dienen soll. Der zweite Parameter legt als Ausgabegerät eine Datei fest. Sinn der Sache: Was Sie jetzt über die Tastatur eingeben, landet in der angegebenen Datei.

Wenn Sie [Return] gedrückt haben, rückt der Cursor an den Anfang der nächsten Zeile. Nun können Sie den Text eintippen. Jede Zeile wird mit [Return] abgeschlossen. Ist der Text komplett, so drücken Sie [F6], was der Tastenkombination [Control-Z] entspricht, und anschließend [Return]. Damit wird das Dateiende markiert.

Bei dieser Form der Texteingabe sind die Korrekturmöglichkeiten leider begrenzt. Nur Fehler in der aktuellen Zeile können verbessert werden. Und auch dies nur durch [Backspace] oder [Pfeil nach links], wobei jeweils der ganze Text bis zum Fehler wieder gelöscht wird.

(Benny Kleinke/tr)

Pfade erweitern

Wollen Sie einen bereits gesetzten Pfad erweitern, so ist normalerweise der gesamte Pfad samt der gewünschten Erweiterung neu einzugeben. Es gibt aber die Alternative, zu diesem Zweck eine kurze Batch-Datei (zum Beispiel »addpath.bat«) zu schreiben. Diese Batch-Datei enthält eine einzige Zeile:

```
path=%path%1;
```

Der Aufruf erfolgt dann mit

```
addpath <Pfaderweiterung>
```

Die Variable »%path%« enthält immer den aktuellen Pfad. Eine Kleinigkeit, die leider in den meisten MS-DOS-Dokumentationen nicht erwähnt wird.

Ein Beispiel: Sie haben eine Festplatte mit den beiden Unterverzeichnissen »c: \ dos« und »c: \ tools«. Ihr Pfad sieht so aus: »path c: \ dos;c: \ tools;«. Nun möchten Sie kurzfristig das Subdirectory »c: \ spiele« an den Pfad anhängen. Wenn Sie die oben genannte Batch-Datei »addpath.bat« angelegt haben, geben Sie ein:

```
addpath c: \ spiele
```

(Benny Kleinke/tr)

Ausgeben von Leerzeilen in Batch-Dateien

Manchmal möchte man bei Textausgaben in Batch-Dateien auch Leerzeilen auf dem Bildschirm darstellen. Gibt man allerdings nur »echo« ein, so erhält man von MS-DOS an dieser Stelle die Information darüber, ob die Echo-Funktion gerade ein- oder ausgeschaltet ist. Eine Leerzeile läßt sich damit leider nicht erzeugen.

Bei MS-DOS Version 2.11 kann das Problem durch ein angehängtes Leerzeichen an den Echo-Befehl gelöst werden, also »echo «. Aber bei späteren Versionen wird leider das

Leerzeichen nach dem Echo-Befehl ignoriert. Hier gibt es aber die Alternative, ein anderes Zeichen direkt an den Echo-Befehl anzuschließen. Ein [Alt-255] ist hier denkbar, aber nicht alle Editoren lassen solche Sonderzeichen zu. [Alt-255] wird von einigen Editoren sogar wieder als Leerzeichen interpretiert. Man gibt in diesem Fall einfach »echo.« ein. Der Punkt funktioniert bei allen Editoren, ist leicht einzugeben und bewirkt die Ausgabe einer Leerzeile. Beachten Sie aber, daß zwischen dem Echo-Befehl und dem Punkt kein Leerzeichen stehen darf. Ansonsten würde an dieser Stelle im Batch-Programm keine Leerzeile, sondern einfach ein Punkt ausgegeben.

Wenn Sie allerdings eine Batch-Datei erzeugen möchten, die auf allen MS-DOS-Versionen lauffähig ist, müssen Sie ein Leerzeichen mit anschließendem [Control-H] – das entspricht [Backspace] – einfügen, was wiederum nicht mit allen Editoren funktioniert. Es muß auf dem Bildschirm »↑H« zu sehen sein. Arbeitet man mit einem Editor, der [Control-H] als [Backspace] interpretiert, so ist es ratsam, statt dessen zum Beispiel einen Stern einzusetzen und diesen mit dem MS-DOS-Programm »debug« oder ähnlichem später durch den Code 08hex zu ersetzen.

(Benny Kleinke/tr)

Aufruf von Batch-Dateien als Unteroutine

Ruft man innerhalb einer Batch-Datei eine weitere Batch-Datei als Unterprogramm auf, so befindet man sich nach Abarbeitung dieser zweiten Datei wieder auf der MS-DOS-Kommandoebene. Der Rest der ersten Batch-Datei wird nicht mehr ausgeführt. In Basic ausgedrückt: Statt eines »gosub« entspricht dies einem »goto«. Dem ist zu begegnen, indem man die zweite Datei mit » \ command /c <Name der zweiten Batch-Datei>« aufruft.

Die MS-DOS-Eingabeebene, also der Kommandointerpreter ist nämlich auch nur ein ganz »normales« Programm, das automatisch geladen wird, wenn Sie Ihren Rechner einschalten. Und als derartiges Programm können Sie es auch später noch starten. Die Option »/c« hinter dem Programmaufruf sagt dem Kommandointerpreter, daß er nach dem Starten eine Batch-Datei ausführen soll. Und zwar die, die hinter der Option angegeben wird.

Ganz einfach wird es bei MS-DOS 3.3: Dort gibt es den Befehl »call«, mit dem eine andere Batch-Datei als Unterprogramm aufgerufen werden kann (also zum Beispiel »call batch.bat«).

(Benny Kleinke/tr)

Definition von Funktionstasten

Unter MS-DOS sind gewöhnlich nur die Funktionstasten [F1] bis [F5] durch den Kommandoeditor belegt (näheres hierzu finden Sie im Handbuch zu Ihrem Computer). Die restlichen Funktionstasten kann man demnach sinnvoll mit häufig benötigten MS-DOS-Kommandos belegen. Dazu muß der Bildschirmtreiber »ansi.sys« in die Config.sys-Datei eingebunden sein (»device = ansi.sys«). Zum Definieren von Funktionstasten werden sogenannte »Escape-Sequenzen« verwendet – Kombinationen aus einem Escape-Zeichen (ASCII 27) und einem nachfolgenden

Kommando. Sobald eine solche Escape-Sequenz auf dem Bildschirm ausgegeben wird (zum Beispiel über einen Type-Befehl), führt der Ansi-Treiber die jeweilige Funktion aus.

Da es sehr schwierig ist, eine Escape-Sequenz in eine Datei einzubinden, empfiehlt sich die Methode, sie vorübergehend in die Prompt-Meldung einzubauen, und das Prompt dann einfach auf dem Bildschirm kurz darstellen zu lassen. Dies wird mittels eines aufeinanderfolgenden »echo on« »echo off« realisiert:

```
echo off
prompt=$e[0;64;"dir";13p
echo on
echo off
prompt=$p$g
```

Diese Batch-Datei belegt die Taste [F6] mit dem Befehl »dir«. Die Zahl nach dem ersten Semikolon – in diesem Beispiel die Zahl 64 – gibt die zu belegende Taste an. Hierbei entspricht

```
64 - [F6]
65 - [F7]
66 - [F8]
67 - ...
```

Sie können auch andere Tasten hierdurch umbelegen, was allerdings nur als Aprilscherz Bedeutung haben dürfte. Die 13 am Ende der Ansi-Sequenz bewirkt, daß nach der Zeichenfolge »dir« automatisch ein »Return« folgt. Der Befehl »echo on« und gleich anschließend »echo off« dient wie bereits gesagt dazu, das Prompt einmal auf dem Bildschirm auszugeben. Da es sich aber hier um eine Ansi-Sequenz zur Tastaturdefinition handelt, sind die Befehle nach dem Prompt-Befehl nicht sichtbar. Die letzte Zeile bewirkt, daß wieder eine normale Prompt-Meldung gesetzt wird. Hier sollten Sie die gewohnte Sequenz, wie sie meist in der »autoexec.bat« verankert ist, wieder einsetzen.

(Benny Kleinke/tr)

Inhalt des Boot-Sektors

Die nachfolgend beschriebene Batch-Datei »readboot.bat« (Listing 1) ist nützlich, um den Inhalt des Boot-Sektors zu lesen und auf dem Bildschirm darzustellen. Das Programm benötigt die beiden MS-DOS-Programme »debug« und »find«, um den Boot-Sektor von der angegebenen Dis-

kette zu lesen. Es speichert den Inhalt des Boot-Sektors in einer Datei zwischen, bevor es ihn auf dem Bildschirm ausgibt.

Dieses Programm kann die Bootsektoren der Laufwerke A bis C lesen. Weitere Laufwerke lassen sich durch einfaches Erweitern der If-Abfragen am Programmanfang in das Programm einbinden.

(Benny Kleinke/tr)

Ändern der Diskettenlaufwerksparameter

Manchmal stimmen die im BIOS-ROM gespeicherten Laufwerksparameter nicht mit den verwendeten Laufwerken überein. Auf diesen Umstand deuten dann beispielsweise häufig auftretende Lesefehler oder starke Geräuschentwicklung beim Diskettenzugriff hin. Sofern diesen Problemen falsche Parameter zugrunde liegen, kann man sich mit folgendem Trick helfen:

Der Interrupt-Zeiger 1Eh enthält die Adresse des Blocks, der die Laufwerksparameter enthält (FDPB). Bei den DOS-Versionen 2.0, 2.1, 3.0 und 3.1 ist diese Adresse 0:522. Der FDPB-Block ist folgendermaßen aufgebaut:

Offset	Funktion	Konfigurierbar	Beispiel
00h	Schrittrate	ja	DF
01h	S/L-Kopf-Ansprechdauer	nein	02
02h	Motoreinschaltverzögerung	nein	25
03h	Zahl der Bytes pro Sektor	nein	02
04h	Nummer des letzten Sektors	nein	09
05h	Lückenbreite	nein	2A
06h	Datenspurlänge	nein	FF
07h	Formatlückenlänge	nein	50
08h	Formatbyte	nein	F6
09h	S/L-Kopfpositionierdauer	ja	0F
0Ah	Startdauer des Motors	ja	02

Schrittratenwerte: FF = 1 ms, EF = 2 ms, DF = 3 ms, und so weiter

Wenn das Laufwerk zu schnell von Spur zu Spur positioniert, entstehen Fehler. Ist die Schrittrate hingegen zu klein, bringt das Laufwerk beängstigende Geräusche hervor. In diesen Fällen muß man zunächst durch Verwenden von Debug herausfinden, welche Parameter momentan gesetzt sind:

```
debug (Starten von Debug)
d 0:78 1 4 (Adresse des FDPB-Blocks herausfinden)
0000:0070 22 05 00 00 (So sieht in den meisten Fällen die
                        Antwort aus)
d 0:522 1 a (Ermitteln der momentanen Parameter)
0000:0520 DF 02 25 02 (so könnten zum Beispiel die aktuellen
09 2A FF 50 Parameter aussehen)
F6 0F 02
q (Vorläufiges Beenden von Debug)
```

Die einfachste Methode, die richtigen Werte herauszufinden, dürfte das schrittweise Ändern der aktuellen Werte sein. In den meisten Fällen reicht es aus, nur den Wert der Schrittrate zu verändern. Neue Werte lassen sich am besten mit einem kleinen Programm ausprobieren. Erzeugen Sie eine Datei »dskfixer.com« mit folgendem Inhalt:

```
n dskfixer.com
a 100
xor ax,ax
mov ds,ax
lds si,[0078]
mov byte ptr [si],<aktuelle Schrittrate>
mov byte ptr [si+9],<aktuelle S/L-Kopfpositionierdauer>
mov byte ptr [si+a],<aktuelle Startdauer des Motors>
int 13
```

```
[DA1] 1: echo off
[PN0] 2: cls
[RC7] 3: echo Lesen und Darstellen des Boot-Sektor
s
[YS0] 4: echo.
[LD2] 5: set drv=error
[843] 6: if "%1"=="a:" set drv=0
[G43] 7: if "%1"=="b:" set drv=1
[S43] 8: if "%1"=="c:" set drv=2
[9D4] 9: if "%drv%"=="error" goto err1
[FK4] 10: echo l 0100 %drv% 0 1 >temp.inp
[544] 11: echo d 0100 02ff >>temp.inp
[8P2] 12: echo q >>temp.inp
[MU4] 13: debug <temp.inp >bootrec.out
[XN5] 14: type bootrec.out | find ":" | more
[0W2] 15: del temp.inp >nul
[8L3] 16: del bootrec.out >nul
[5D1] 17: goto end
[7T0] 18: :err1
[YS0] 19: echo.
[2T6] 20: echo Fehler: Ungültige Laufwerksangabe
[S75] 21: echo Format: READBOOT Laufwerk
[YS0] 22: echo.
[ZN0] 23: :end
[SF1] 24: set drv=
```

Listing 1. »readboot.bat«, Boot-Sektor lesen per Batch-Datei

int 20

r cx
17
w
q

Die Datei muß unbedingt die hier nach dem zweiten Int-Befehl abgedruckte Leerzeile enthalten. Noch ein Hinweis: Die vorgestellte Lösung sollten Sie nur dann anwenden, wenn Ihnen Begriffe wie »Offset«, »FDPB-Block« oder der Umgang mit dem Debugger keine Probleme bereiten.

Nun »debug < diskfixer.inp« aufrufen, wodurch automatisch das Programm »dskfixer.com« erzeugt wird.

Beschreibung: Das Programm liest die Adresse des FDPB-Blocks aus dem Vektor 1Eh. Anschließend ändert es die Werte in dem FDPB-Block, ruft den Interrupt 13 auf, dessen AH-Register immer noch gleich 0 ist, um die Parameter zu installieren. Das Programm endet über den Interrupt 20.

Rufen Sie Dskfixer auf, und benutzen Sie »type«, um probeweise irgendeine Datei von der Diskette zu lesen. Ist das Problem schlimmer geworden, ändern Sie die Parameter genau umgekehrt, und versuchen es erneut. Dieses Verfahren wird solange angewendet, bis entweder das Problem gelöst ist, oder die Parameter so stark verändert sind, daß man sicher sein kann, daß verkehrte Parameter nicht die Ursache des Fehlers sind.

Ist der Fehler behoben, muß »dskfixer.com« nur noch in die Autoexec-Datei eingebunden werden, und die Probleme mit dem Laufwerk sind ein für allemal beseitigt.

(Benny Kleinke/tr)

Unterdrücken von Batch-Kommandos

Die MS-DOS-Version 3.3 erlaubt es, das Anzeigen einer Zeile in einer Batch-Datei durch Voranstellen eines »@« (Klammeraffe) zu verhindern. Eine Anwendung wäre zum Beispiel das Unterdrücken der Ausgabe von »echo off« am Anfang einer Batch-Datei:

```
@echo off
rem Das ist ein Test
```

Beim Abarbeiten dieser Datei würde nichts am Bildschirm angezeigt. Bei früheren Versionen funktioniert dies leider noch nicht. Hier muß man sich mit einem Trick behelfen. Entweder man löscht gleich nach dem Echo-off-Befehl mit »cls« den gesamten Bildschirm oder verwendet folgende elegantere Methode:

```
echo off
echo ^[[s^[[1A^K^[[u
```

Hierbei ist zu beachten, daß sowohl der Bildschirmtreiber »ansi.sys« in die Datei »config.sys« eingebunden sein muß, als auch, daß es sich bei dem Zeichen »^[[« um die Tastenkombination <Ctrl-]> (Alt-27) handelt.

Das Programm arbeitet auf folgende Weise:

```
^[[s   Speichert die aktuelle Cursorposition.
^[[1A  Bewegt den Cursor eine Zeile nach oben.
^K     Löscht die Zeichen vom Cursor bis zum Ende der Zeile.
^[[u   Setzt den Cursor wieder auf die ursprüngliche Position.
```

Hierdurch wird »echo off« zwar auf dem Bildschirm ausgegeben, aber sofort wieder gelöscht, bevor man es lesen kann.

(Benny Kleinke/tr)

Wozu dient »driver.sys«?

Hat der PC zwei Diskettenlaufwerke, so ist es normalerweise nicht möglich, von einer Diskette eine Kopie anzufertigen, ohne Quell- und Zieldiskette auf beide Laufwerke zu verteilen. Hierfür kann man aber die ab Version 3.0 von MS-DOS mitgelieferte Datei »driver.sys« verwenden. Möchte man beispielsweise Laufwerk A als Laufwerk C

ansprechen, kann man in der Config.sys-Datei folgende Zeile einbinden:

```
device=driver.sys /c:0 /t:80 /s:15 /h:2 /c /f:1
```

Die Bedeutung der Parameter zeigen wir Ihnen anhand einer Tabelle:

Parameter	Wirkung
/c:0	Das Laufwerk 0 (normalerweise A) soll nun C heißen
/t:80	Anzahl der Spuren auf diesem Laufwerk (bei 360 KByte-Laufwerken sind es 40 Spuren anstatt 80)
/s:15	Anzahl Sektoren per Spur
/h:2	Anzahl der Schreib/Leseköpfe im Laufwerk
/c	Läßt den Rechner offene Laufwerksklappen erkennen (nur bei 3,5-Zoll-Laufwerken)
/f:1	Laufwerkstyp (0 = 180/360 KByte, 1 = 1,2 MByte, 2 = 3,5-Zoll)

(Benny Kleinke/tr)

Format der Datums- und Zeitangabe

Im Normalzustand, also beispielsweise wenn man den Computer bootet, ohne über eine Config.sys-Datei auf der Systemdiskette zu verfügen, ist das amerikanische Zeit- und Datumsformat voreingestellt. Das bedeutet, daß man sich mit einer Zwölfstundenanzeige und der Reihenfolge Monat vor Tag herumzuschlagen hat.

Hier läßt sich allerdings leicht Abhilfe schaffen: Man muß lediglich in die Config.sys-Datei die Zeile »country = 049« einbinden. Dies bewirkt die Umstellung des Datum- und Zeitformats. Ferner wird als Dezimaltrennzeichen das Komma (statt im Amerikanischen ein Punkt) und als Währungssymbol »DM« eingebunden. Daß die anzugebende Nummer für Country gerade 049 ist, was der Vorwahl der Bundesrepublik entspricht, ist kein Zufall. In den meisten Fällen entspricht diese Länderkennzahl der jeweiligen Vorwahl, was die Konfiguration erleichtert.

Um bei der MS-DOS-Version 3.3 diese Zeile verwenden zu können, muß im Hauptverzeichnis (Root-Directory) die Datei »country.sys« vorhanden sein. Soll diese in ein Subdirectory (zum Beispiel »c: \dos«) gelegt werden, so muß die Country-Angabe erweitert werden:

```
country=049,850,c: \dos \country.sys
```

Die Zahl 850 wählt die europäische Zeichensatztabelle aus, die normalerweise automatisch eingeschaltet ist.

(Benny Kleinke/tr)

Der Break-Befehl

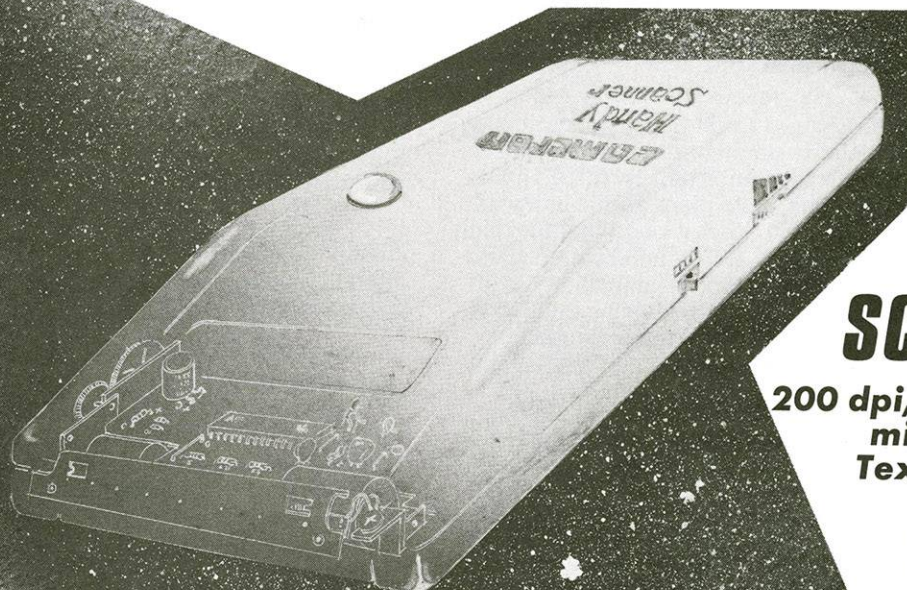
Eigentlich funktioniert die Tastenkombination [Ctrl-Break] nur während Tastatureingaben und Bildschirm-, Drucker- oder sonstigen Schnittstellenausgaben. Stellt man fest, daß ein Programm noch erheblich längere Zeit rechnen wird, ist man versucht, es mit [Ctrl-Break] abubrechen. Dieser Tastendruck wird brav registriert, jedoch nicht weiter bearbeitet. Erst wenn die Berechnung abgeschlossen ist, wird das Programm abgebrochen. Ähnliche Probleme ergeben sich bei der Ausführung von Batch-Dateien.

Hiergegen hilft der Befehl »break on«. Direkt von der Tastatur eingegeben oder in einer Batch-Datei enthalten, bewirkt er, daß, selbst wenn gerade keine Aus- oder Eingabe stattfindet, der Rechner sofort auf [Ctrl-Break] reagiert. Natürlich wird der Rechner durch diese Option minimal verlangsamt. Durch den Befehl »break off« können Sie wieder in den normalen Abfragemodus übergehen. Durch die Eingabe von »break« alleine läßt sich der aktuelle Modus abfragen. Außerdem können Sie diese Option gleich beim Systemstart aktivieren, indem Sie in die Config.sys-Datei die Zeile »break = on« einfügen.

(Benny Kleinke/tr)

TEXT + GRAFIK

für IBM PC / XT / AT, Atari ST und Commodore AMIGA



Komplett-
system

HANDY SCANNER

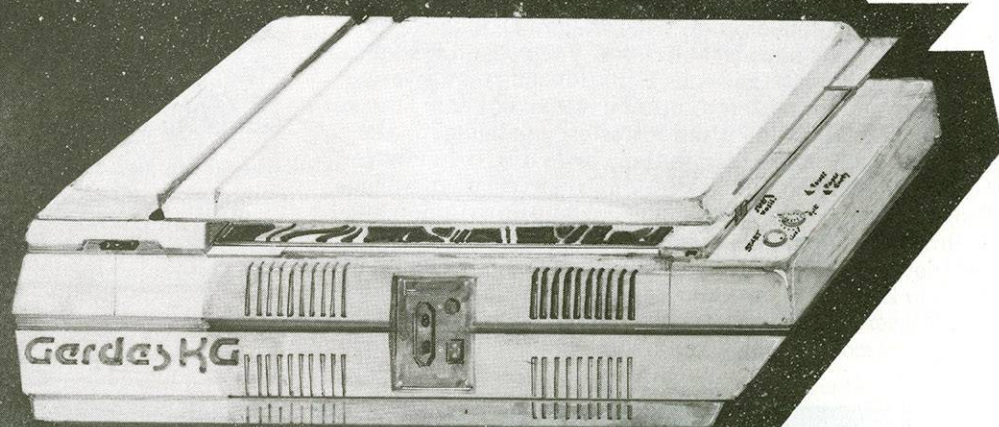
200 dpi, 64 mm breit
mit Grafik- und
Texterfassungs-
Software

s/w-Version:

DM 598,-

Graustufen-
Version:

DM 898,-



Komplettsystem

PERSONAL A4 SCANNER

DIN A 4 Flachblattscanner
200 dpi
mit Text- und Grafikerfassungs-Software
DM 1698,-

Info gratis.

Händlerkonditionen bei Gewerbenachweis

Imperial SOFTWARE Systems Gerdes KG

Rochuscenter
Lessenicher Straße 9
D-5300 Bonn 1
Tel.: 02 28 / 61 62 10
Tel.: 02 28 / 25 24 74
Fax: 02 28 / 61 62 29
BTX: 02 28 / 61 62 10

Editor in GW-Basic

Mit dem Programm »bukamav.asc« verfügen Sie nicht nur über eine Textverarbeitung, Sie können auch Unterverzeichnisse anlegen und löschen, Disketten formatieren und und und ...

Wenn Sie das Basic-Programm »bukamav.asc« starten, erscheint als erstes eine kleine Benutzeroberfläche auf dem Bildschirm, die in vier Fenster unterteilt ist (Bild 1). Das größte Fenster auf der linken Seite dient zum Auswählen der verschiedenen Funktionen. Rechts oben befindet sich ein Window, das bei Fehleingaben die Fehlermeldungen und Warnhinweise ausgibt. Das Fenster darunter erteilt Ihnen zu den einzelnen Funktionen Hinweise und Erklärungen, während das letzte Window für die Eingabe dient. Außerdem ist im Hauptmenü eine Uhr installiert.

Das Programm bietet Ihnen im Funktionenmenü insgesamt 18 Punkte an, so daß Sie das Programm nicht verlassen müssen, wenn Sie beispielsweise vergessen haben, die Diskette zu formatieren.

Die ersten vier Punkte beziehen sich auf den Editor. Hier können Sie wählen, ob Sie eine Textdatei lesen, eine neue Datei eröffnen, verschiedene Dateien verbinden oder zu einer bereits bestehenden Datei Text anfügen wollen. Außerdem können Sie eine bereits vorhandene Datei duplizieren. Das Programm erlaubt, die ersten sechs Funktionstasten mit beliebigen Zeichen zu versehen (Bild 2). Sie können dann mit diesen Funktionstasten im Editiermodus sehr einfach Rahmen zeichnen oder Sonderzeichen eingeben. Die nächsten drei Funktionen erlauben das Kopieren, Löschen und Umbenennen von Dateien. Darauf folgen Funktionen, die hauptsächlich mit Verzeichnissen zu tun haben. Mit den letzten sechs Funktionsaufrufen können Sie Disketten formatieren, Ihre Textdatei auf dem Drucker ausgeben lassen, die Funktionstasten neu belegen, die Größe einer Textdatei feststellen sowie die Hilfefunktion anwählen. Mit dem letzten Menüpunkt verlassen Sie das Programm. Zum Anwählen dieser Hauptfunktionen steht es Ihnen frei, entweder die Pfeiltasten, die Leertaste, die Tabulator- oder die Backspace-Taste zu verwenden. Außerdem können Sie mit der Pgup- und Pgdn-Taste innerhalb dieser Hauptfunktionen schneller »herumspringen«.

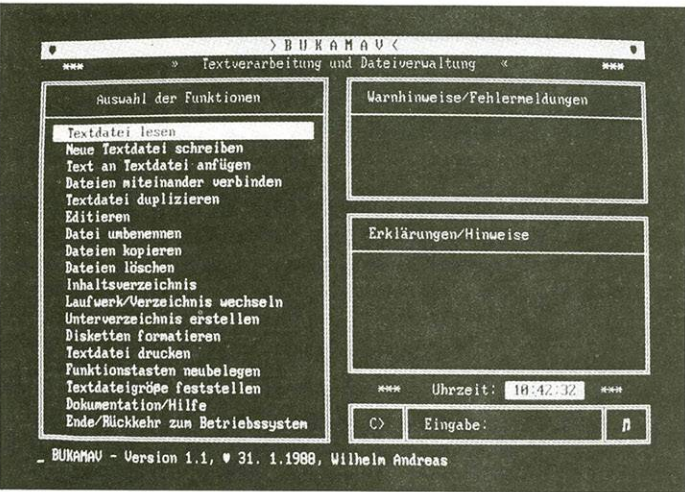


Bild 1. Das Hauptmenü von »bukamav.asc«

Falls Sie bei der Eingabe einen Fehler gemacht haben, werden Sie von dem Fenster »Warnhinweise/Fehlermeldungen« darauf aufmerksam gemacht. Das darunterliegende Fenster mit dem Namen »Erklärungen/Hinweise« versorgt Sie mit nötigen Zusatzinformationen. Haben Sie beispielsweise den Punkt »Disketten formatieren« angewählt, erscheinen in diesem Fenster die Erklärungen und Hinweise darüber, was Sie eingeben müssen, wenn Sie eine 180 KByte-, eine 360KByte- oder eine 1,2MByte-Diskette formatieren wollen. Sie können bei Bedarf auch den Diskettennamen ändern. Auch wenn Sie den Menüpunkt »Unterverzeichnis erstellen« anwählen, stehen Ihnen mehrere Unterfunktionen zur Verfügung. Hier lassen sich Unterverzeichnisse anlegen, umbenennen oder löschen und natürlich auch anzeigen.

Das Programm »bukamav.asc« ist vollkommen menügesteuert und deshalb leicht bedienbar

Wenn Sie die Funktionstaste [F1] betätigen, erhalten Sie in der Überschriftszeile das aktuelle Verzeichnis. Die Taste [F1] ist als Schalter programmiert, so daß Sie mit erneuter Betätigung die Anzeige wieder abschalten können. Den Namen der aktuellen Textdatei sowie die Datei, die Sie für den Druck vorgesehen haben, erhalten Sie mit der Taste [F2]. Mit den Funktionstasten 3 und 4 können Sie sich den voreingestellten Datei- und Pfadnamen anzeigen lassen. Die Warntöne bei Falscheingaben werden mit der Taste [F5] eingeschaltet und mit [F6] wieder deaktiviert. Mit Hilfe der Funktionstaste [F10] löschen Sie die Anzeigen von [F2], [F3] und [F4] wieder.

Bei Aufruf der Funktion »Dokumentation/Hilfe« stellt Ihnen »bukamav.bas« fünf Bildschirmseiten mit den wichtigsten Erklärungen zur Verfügung. Dies reicht auch vollkommen aus, da das Programm menügesteuert ist und bei Falscheingaben entsprechende Fehlermeldungen ausgibt. Außerdem versorgt Sie das Fenster »Erklärungen/Hinweise« mit den nötigen Informationen, so daß man sich schnell in die Textverarbeitung eingearbeitet hat.

Wenn Sie sich in Basic schon eine Arbeitsumgebung programmiert haben, die Ihre individuellen Anforderungen erfüllt, und es fehlt Ihnen zur Abrundung noch eine Textverarbeitung, dann ist »bukamav.asc« genau das richtige für Sie!

(Wilhelm Andreas/gh)

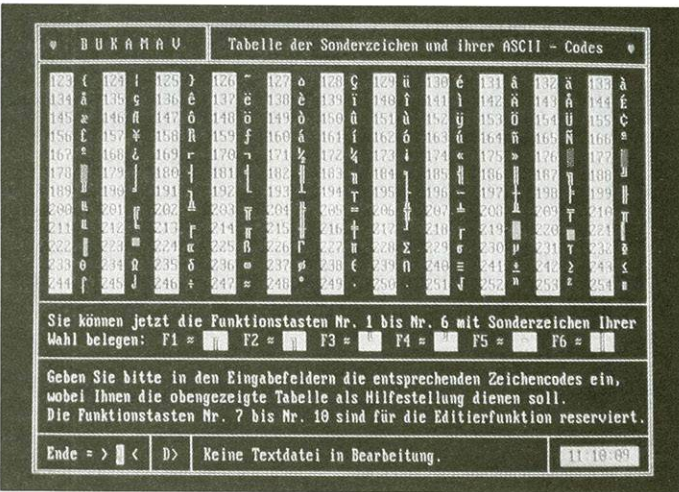


Bild 2. Das Untermenü zum Belegen der Funktionstasten


```

[568] 10 *****
[VC7] 20 *
      * > B U K A M A V <
      *
[968] 30 *****
[9R9] 40 *
      * Textprogramm und Dateiver
      * waltung
[2B6] 50 *
      *
[GWB] 60 * Version 1.1 für GW-BASIC, 31. 1.1988, gesch
      * rieben auf Commodore PC 10-II *
[00B] 70 * Copyright (c) by Wilhelm Andreas, Im Östern
      * 6, 4690 Herne 2 *
[5C6] 80 *
      *
[N68] 90 *****
[0H0] 100 '
[3H0] 110 '
[ED5] 120 REM Programm initialisieren
[9H0] 130 '
[XD3] 140 ON ERROR GOTO 11860
[6P3] 150 ON TIMER(1) GOSUB 8110
[2B7] 160 OPTION BASE 0:DIM F$(20),MN$(32),TEXT$(50),A$(
      * 12)
[P58] 170 FOR A = 1 TO 32:READ MN$(A):NEXT:LET L$ = SPA
      * CE$(80)
[H28] 180 SCREEN 0,0,0:WIDTH 80:VIEW PRINT:WIDTH LPRINT
      * 80
[6P5] 190 CLS:LOCATE ,0:KEY OFF:GOSUB 11400
[C54] 200 LOCATE 2,8:SHELL "CHDIR"
[4B4] 210 IF SCREEN(2,8) <> 32 THEN 230
[CR7] 220 LW$ = CHR$(SCREEN(3,1)) + ">":LET DOSALT = 1:
      * GOTO 240
[YV4] 230 LET LW$ = CHR$(SCREEN(2,8)) + ">"
[HY3] 240 LET MX = 21:LET MY = 63
[HH0] 250 '
[MH0] 260 '
[1B6] 270 REM Aufruf der Konfigurationsdatei
[UH0] 280 '
[A64] 290 OPEN "I",#1,"BUKAMAV.NED"
[CYA] 300 INPUT #1,F1:INPUT #1,F2:INPUT #1,F3:INPUT #1,
      * F4:INPUT #1,F5:INPUT #1,F6
[FF6] 310 FOR A = 1 TO 20:LINE INPUT #1,F$(A):NEXT
[KN2] 320 CLOSE:GOTO 500
[DH0] 330 '
[GH0] 340 '
[KDA] 350 REM Systemeinstellung bei fehlender Konfigura
      * tionsdatei
[PH0] 360 '
[KL5] 370 LET F1 = 201:LET F2 = 187:LET F3 = 200
[AP5] 380 LET F4 = 188:LET F5 = 205:LET F6 = 186
[8EB] 390 LET F$(1) = "BUKAMAV.BUK":LET F$(2) = "NOTIZE
      * N.BUK":LET F$(3) = "BRIEF.BUK"
[AE9] 400 LET F$(4) = "TAGEBUCH.BUK":LET F$(5) = ".TXT"
      * :LET F$(6) = ".MSG"
[M85] 410 LET F$(7) = ".DAT":LET F$(8) = ".BAT"
[W58] 420 LET F$(9) = "AUTOEXEC.BAT":LET F$(10) = "BUKA
      * MAV.NED"
[BAA] 430 LET F$(11) = "A:\":LET F$(12) = "B:\":LET F$(
      * 13) = "C:\":LET F$(14) = "D:\":
[2AA] 440 LET F$(15) = "\BUKAMAV":LET F$(16) = "\TEXTE"
      * :LET F$(17) = "\BRIEF"
[PX7] 450 LET F$(18) = "\BASIC":LET F$(19) = ".":LET F
      * $(20) = "\"
[SH0] 460 '
[VH0] 470 '
[BG3] 480 REM Menü aufbauen
[1K0] 490 '
[7P6] 500 CLS:LET MDAZG = 0:LET AZG = 0:LET MFN = 0
[3PA] 510 COLOR 8,7:PRINT " ";CHR$(3);TAB(32) MN$(19);T
      * AB(79) CHR$(3);" ";COLOR 7,0
[EX7] 520 GOSUB 7690:PRINT CHR$(201);STRING$(37,205);CH
      * R$(187)
[ZT5] 530 FOR A = 1 TO 20:PRINT CHR$(186):NEXT
[F68] 540 LOCATE 24,1:PRINT CHR$(200);STRING$(37,205);C
      * HR$(188);
[RN8] 550 FOR A = 23 TO 4 STEP -1:LOCATE A,39:PRINT CHR
      * $(186):NEXT
[5FA] 560 LOCATE 4,9:COLOR 10,0:PRINT "Auswahl der Funk
      * tionen":COLOR 7,0
[UW7] 570 LOCATE 5,1:PRINT CHR$(199);STRING$(37,196);CH
      * R$(182)
[T18] 580 LOCATE 3,42:PRINT CHR$(201);STRING$(37,205);C
      * HR$(187)
[2N7] 590 FOR A = 4 TO 9:LOCATE A,42:PRINT CHR$(186):NE
      * XT
[E98] 600 LOCATE 10,42:PRINT CHR$(200);STRING$(37,205);
      * CHR$(188)

```

```

[0T8] 610 FOR A = 9 TO 4 STEP -1:LOCATE A,80:PRINT CHR$(
      * 186):NEXT
[1GB] 620 LOCATE 4,45:COLOR 10,0:PRINT "Warnhinweise/Fe
      * hlermeldungen":COLOR 7,0
[638] 630 LOCATE 5,42:PRINT CHR$(199);STRING$(37,196);C
      * HR$(182)
[Y98] 640 LOCATE 11,42:PRINT CHR$(201);STRING$(37,205);
      * CHR$(187)
[5T7] 650 FOR A = 12 TO 19:LOCATE A,42:PRINT CHR$(186):
      * NEXT
[1A8] 660 LOCATE 20,42:PRINT CHR$(200);STRING$(37,205);
      * CHR$(188)
[UZ8] 670 FOR A = 19 TO 12 STEP -1:LOCATE A,80:PRINT CH
      * R$(186):NEXT
[LAA] 680 LOCATE 12,45:COLOR 10,0:PRINT "Erklärungen/Hi
      * nweise":COLOR 7,0
[XC8] 690 LOCATE 13,42:PRINT CHR$(199);STRING$(37,196);
      * CHR$(182)
[YY9] 700 LOCATE 21,46:COLOR 10,0:PRINT "*** Uhrzeit
      * : ***":COLOR 7,0
[4B7] 710 LOCATE 21,62:PRINT STRING$(10,219):LOCATE 22,
      * 42
[H2A] 720 PRINT CHR$(201);STRING$(6,205);CHR$(203);STRI
      * NG$(24,205);CHR$(203);
[GT6] 730 PRINT STRING$(5,205);CHR$(187):LOCATE 23,42
[K7A] 740 PRINT CHR$(186);TAB(49) CHR$(186);TAB(74) CHR
      * $(186);TAB(80) CHR$(186);
[5NA] 750 LOCATE 24,42:PRINT CHR$(200);STRING$(6,205);C
      * HR$(202);STRING$(24,205);
[896] 760 PRINT CHR$(202);STRING$(5,205);CHR$(188);
[CA8] 770 LOCATE 23,52:COLOR 10,0:PRINT "Eingabe:";GOS
      * UB 7750
[2L7] 780 FOR A = 1 TO 18:LOCATE A + 5,3:PRINT MN$(A):N
      * EXT
[T49] 790 LET XT = 21:LET YT = 63:LOCATE 23,45:COLOR 10
      * ,0:PRINT LW$;
[927] 800 GOSUB 11660:ON RM GOTO 1200,1360,1990,3090,35
      * 50,4090,4420
[HH0] 810 '
[MH0] 820 '
[FY3] 830 REM Textdatei lesen
[UH0] 840 '
[6XA] 850 COLOR 8,7:GOSUB 7440:LET AM = 1:GOSUB 7820:ON
      * R GOTO 1200,6870,3550,3860
[FG4] 860 IF A$ = "A" OR A$ = "a" THEN 940
[6X9] 870 IF A$ = "N" OR A$ = "n" THEN LET LESEDATEI$ =
      * NEUERNAME$:GOTO 950
[ZX9] 880 IF A$ = "R" OR A$ = "r" THEN LET LESEDATEI$ =
      * "RÖDEL.NED":GOTO 950
[KT9] 890 IF A$ = "S" OR A$ = "s" THEN LET LESEDATEI$ =
      * "GASED.NED":GOTO 950
[YU9] 900 IF A$ = "T" OR A$ = "t" THEN LET LESEDATEI$ =
      * "TIMOP.NED":GOTO 950
[L29] 910 IF A$ = "Z" OR A$ = "z" THEN LET LESEDATEI$ =
      * ZIEL$:GOTO 950
[M96] 920 IF A$ = CHR$(13) THEN GOSUB 8510 ELSE 850
[VDB] 930 IF EINGABES$ <> "" THEN LET DATEINAME$ = EINGA
      * BES$ ELSE GOSUB 10200:GOTO 850
[LW4] 940 LET LESEDATEI$ = DATEINAME$
[0G5] 950 OPEN "I",#1,LESEDATEI$:LET SNR = 0
[992] 960 LET MFN = 0
[NC5] 970 CLS:LET Z = 0:LET SNR = SNR + 1
[ZE4] 980 IF EOF(1) = -1 THEN GOTO 1000
[N5C] 990 LET Z = Z + 1:LINE INPUT #1,TEXT$:LOCATE Z,1:
      * PRINT TEXT$;IF Z < 24 THEN 980
[TP7] 1000 IF EOF(1) = -1 AND SNR = 1 THEN CLOSE:GOTO 1
      * 0280
[A5B] 1010 LOCATE 25,1:COLOR 8,7:PRINT " ";CHR$(3);" >
      * System wartet < Ende = > ";
[VEA] 1020 COLOR 16,7:PRINT "Q";:COLOR 8,7:PRINT " <
      * ";IF EOF(1) = -1 THEN 1050
[UCB] 1030 PRINT "Vor = > ";:COLOR 16,7:PRINT "Leertast
      * e";:COLOR 8,7:PRINT " < ";
[6G3] 1040 IF SNR = 1 THEN 1060
[TUA] 1050 PRINT "Zurück = > ";:COLOR 16,7:PRINT "TAB";
      * :COLOR 8,7:PRINT " < ";
[DK5] 1060 PRINT TAB(79) CHR$(3);" ";:COLOR 7,0
[SH4] 1070 LET A$ = INKEY$:GOSUB 8060
[KC5] 1080 IF A$ = " " AND NOT EOF(1) THEN 970
[X06] 1090 IF A$ = CHR$(13) AND NOT EOF(1) THEN 970
[WBA] 1100 IF A$ = "Q" OR A$ = "q" OR A$ = CHR$(0) + CH
      * R$(15) THEN CLOSE:GOTO 10280
[LD6] 1110 IF A$ <> CHR$(8) AND A$ <> CHR$(9) THEN 1070
[TNA] 1120 IF SNR = 1 THEN 1070 ELSE LOCATE 25,1:PRINT
      * STRING$(30,176);CHR$(16);
[7FB] 1130 COLOR 18,7:PRINT MN$(22);:COLOR 7,0:PRINT CH
      * R$(17);STRING$(30,176);:CLOSE
[RHA] 1140 LET SNR = SNR - 2:LET ZNR = SNR * 24:OPEN "I
      * ",#1,LESEDATEI$:LET Z = 0

```

Das Basic-Listing »bukamav.asc« ist nicht nur ein Texteditor


```

[ZR9] 1150 WHILE Z < ZNR:LET Z = Z + 1:LINE INPUT #1,TE
XT$:WEND:GOTO 970
[KN0] 1160 '
[PN0] 1170 '
[NC5] 1180 REM Neue Textdatei schreiben
[YN0] 1190 '
[BKA] 1200 COLOR 8,7:GOSUB 7450:LET AM = 2:GOSUB 7820:O
N R GOTO 1360,850,3860,4090
[ND6] 1210 IF AS = CHR$(13) THEN GOSUB 8510 ELSE 1200
[HVB] 1220 IF EINGABES <> "" THEN LET DATEINAME$ = EING
ABES ELSE GOSUB 10200:GOTO 1200
[XB5] 1230 OPEN "O",#1,DATEINAME$:LET RM = 1
[E18] 1240 CLS:COLOR 8,7:PRINT " ";CHR$(3);TAB(4) MN$(1
9);MN$(2)
[KV8] 1250 GOSUB 9280:GOSUB 11550:GOSUB 10680:GOSUB 112
00:LET NA = 1
[EM7] 1260 IF MA = 1 THEN LOCATE 3,1:PRINT TEXT$:LET MA
= 0
[GH9] 1270 LET NAZ = CSRLIN:IF NAZ = 24 THEN LET NAZ =
23:PRINT CHR$(0);
[DP8] 1280 GOSUB 9360:IF FUNK = 4 THEN CLOSE:LET NA = 0
:GOTO 10280
[E49] 1290 IF FUNK = 1 OR FUNK = 2 OR FUNK = 3 OR FUNK
= 5 THEN 1270
[MD8] 1300 LOCATE AKTZ,1:PRINT TEXT$;SPACE$(80 - LEN(TE
XT$));
[894] 1310 PRINT #1,TEXT$:GOTO 1270
[8N0] 1320 '
[CN0] 1330 '
[ZN5] 1340 REM Text an Textdatei anfügen
[MN0] 1350 '
[3PA] 1360 COLOR 8,7:GOSUB 7460:LET AM = 3:GOSUB 7820:O
N R GOTO 1490,1200,4090,4420
[2V4] 1370 IF AS = "A" OR AS = "a" THEN 1400
[8F6] 1380 IF AS = CHR$(13) THEN GOSUB 8510 ELSE 1360
[BXB] 1390 IF EINGABES <> "" THEN LET DATEINAME$ = EING
ABES ELSE GOSUB 10200:GOTO 1360
[5U3] 1400 OPEN "I",#1,DATEINAME$
[K52] 1410 LET MFN = 0
[X07] 1420 WHILE NOT EOF(1):LINE INPUT #1,TEXT$:WEND
[2Y7] 1430 CLOSE:LET MA = 1:LET RM = 2:OPEN "A",#1,DATE
INAME$
[RK9] 1440 CLS:COLOR 8,7:PRINT " ";CHR$(3);TAB(4) MN$(1
9);MN$(3):GOTO 1250
[RN0] 1450 '
[VN0] 1460 '
[E86] 1470 REM Dateien miteinander verbinden
[3P0] 1480 '
[5RA] 1490 COLOR 8,7:GOSUB 7470:LET AM = 4:GOSUB 7820:O
N R GOTO 1740,1360,4420,4860
[H99] 1500 IF AS = "A" OR AS = "a" THEN LET ZIEL$ = DAT
EINAME$:GOTO 1560
[515] 1510 IF AS = "Z" OR AS = "z" THEN 1560
[544] 1520 IF AS <> CHR$(13) THEN 1490
[FR8] 1530 LOCATE 14,45:COLOR 1,0:PRINT MN$(20);" der Z
ieldatei"
[PR7] 1540 LOCATE 15,45:PRINT "ein:":COLOR 7,0:GOSUB 85
40
[VK8] 1550 IF EINGABES <> "" THEN LET ZIEL$ = EINGABES
ELSE 1690
[AR4] 1560 OPEN "A",#2,ZIEL$:GOSUB 10200
[988] 1570 LOCATE 14,45:COLOR 1,0:PRINT MN$(20);" der 2
. Datei"
[6S7] 1580 LOCATE 15,45:PRINT "ein:":COLOR 7,0:GOSUB 85
40
[3PA] 1590 IF EINGABES <> "" THEN LET DATEI2$ = EINGABE
$ ELSE CLOSE:GOTO 1690
[4X4] 1600 OPEN "I",#1,DATEI2$:GOSUB 10200
[PX9] 1610 LOCATE 14,45:PRINT "Die Textdateien";:LOCATE
15,45:COLOR 8,7
[ZU8] 1620 PRINT " ";ZIEL$; " ";:COLOR 7,0:PRINT " und "
;:COLOR 16,7
[B07] 1630 PRINT " ";DATEI2$; " ";:COLOR 7,0:LOCATE 16,45
[EP6] 1640 PRINT "werden miteinander verbunden."
[EBC] 1650 LOCATE 18,53:COLOR 18,7:PRINT MN$(22):TIMER
ON:GOSUB 11300:CLOSE:TIMER OFF
[3S9] 1660 GOSUB 11720:LOCATE 15,46:COLOR 8,7:PRINT ZIE
L$; " ";:COLOR 7,0
[G0C] 1670 PRINT " und ";:COLOR 8,7:PRINT " ";DATEI2$:C
OLOR 7,0:LOCATE 16,46:PRINT "u"
[PW6] 1680 LOCATE 18,53:PRINT SPACE$(19):GOSUB 10060
[MM3] 1690 GOSUB 10200:GOTO 1490
[CN0] 1700 '
[GN0] 1710 '
[FX4] 1720 REM Textdatei duplizieren
[SN0] 1730 '
[TRA] 1740 COLOR 8,7:GOSUB 7480:LET AM = 5:GOSUB 7820:O
N R GOTO 1990,1490,4860,5310
[XR9] 1750 IF AS = "A" OR AS = "a" THEN LET QUELLE$ = D
ATEINAME$:GOTO 1820

```

```

[C4A] 1760 IF AS = "R" OR AS = "r" THEN LET QUELLE$ = "
RÖDEL.NED":GOTO 1820
[ES9] 1770 IF AS = "S" OR AS = "s" THEN LET QUELLE$ = "
GASED.NED":GOTO 1820
[LW9] 1780 IF AS = "T" OR AS = "t" THEN LET QUELLE$ = "
TIMOP.NED":GOTO 1820
[739] 1790 IF AS = "Z" OR AS = "z" THEN LET QUELLE$ = Z
IEL$:GOTO 1820
[TE6] 1800 IF AS = CHR$(13) THEN GOSUB 8510 ELSE 1740
[0V8] 1810 IF EINGABES <> "" THEN LET QUELLE$ = EINGABE
$ ELSE 1940
[X35] 1820 OPEN "I",#1,QUELLE$:GOSUB 10200
[SR8] 1830 LOCATE 14,45:COLOR 1,0:PRINT MN$(20);" der Z
ieldatei"
[ZR7] 1840 LOCATE 15,45:PRINT "ein:":COLOR 7,0:GOSUB 85
40
[AGA] 1850 IF EINGABES <> "" THEN LET ZIEL$ = EINGABES
ELSE CLOSE:GOTO 1940
[1S4] 1860 OPEN "O",#2,ZIEL$:GOSUB 10200
[RR8] 1870 LOCATE 14,45:PRINT "Von der Textdatei ";:COL
OR 16,7
[087] 1880 PRINT " ";QUELLE$; " ";:COLOR 7,0:LOCATE 15,4
5
[DC9] 1890 PRINT "wird ";MN$(32):LOCATE 16,45:PRINT "Na
men ";:COLOR 8,7
[MG8] 1900 PRINT " ";ZIEL$; " ";:COLOR 7,0:PRINT " angef
fertigt."
[7BC] 1910 LOCATE 18,53:COLOR 18,7:PRINT MN$(22):TIMER
ON:GOSUB 11300:CLOSE:TIMER OFF
[1KB] 1920 GOSUB 11720:LOCATE 14,64:COLOR 8,7:PRINT QUE
LLE$:COLOR 7,0:LOCATE 15,45
[62A] 1930 PRINT "wurde ";MN$(32):LOCATE 18,53:PRINT SP
ACE$(19):GOSUB 10060
[2M3] 1940 GOSUB 10200:GOTO 1740
[6P0] 1950 '
[AP0] 1960 '
[NV2] 1970 REM Editieren
[KP0] 1980 '
[9RA] 1990 COLOR 8,7:GOSUB 7490:LET AM = 6:GOSUB 7820:O
N R GOTO 2870,1740,5310,6150
[5U4] 2000 IF AS = "A" OR AS = "a" THEN 2030
[UE6] 2010 IF AS = CHR$(13) THEN GOSUB 8510 ELSE 1990
[6XB] 2020 IF EINGABES <> "" THEN LET DATEINAME$ = EING
ABES ELSE GOSUB 10200:GOTO 1990
[TB5] 2030 OPEN "I",#1,DATEINAME$:LET RM = 3
[M52] 2040 LET MFN = 0
[CZ3] 2050 OPEN "O",#2,"TIMOP.NED"
[8BA] 2060 GOSUB 11550:CLS:COLOR 8,7:PRINT " ";CHR$(3);
TAB(4) MN$(19); " ";MN$(6)
[T6B] 2070 GOSUB 9280:LOCATE 25,1:PRINT CHR$(4);" Zeile
übernehmen = F9 Zeile
[LKA] 2080 PRINT "einfügen = F7 Zeile löschen = F8
Ende = F10 ";:COLOR 7,0
[Z25] 2090 GOSUB 11200:LET Z = 0:LET MZ = 0
[NN7] 2100 IF EOF(1) = -1 THEN 2310 ELSE LINE INPUT #1,
ETXT$
[DU7] 2110 LET Z = Z + 1:IF Z > 19 THEN LET Z = 20:LET
MZ = 1
[E38] 2120 COLOR 8,7:LOCATE 2 + Z,1:LET Y = CSRLIN:PRIN
T L$;
[34A] 2130 LOCATE 2 + Z,1:PRINT ETXT$;:LOCATE 2 + Z,1:G
OSUB 9360:GOTO 2160
[6G9] 2140 IF EM = 1 THEN LOCATE Y + 1:COLOR 7,0:PRINT
ETXT$;:COLOR 8,7
[RW3] 2150 LOCATE Y,1:GOSUB 9360
[TAA] 2160 IF FUNK = 4 THEN PRINT #2,ETXT$:GOSUB 2670:G
OSUB 11300:GOTO 2500
[H99] 2170 IF FUNK = 5 THEN CLOSE #2:LET RST = 1:GOSUB
2670:GOTO 2820
[5N3] 2180 IF FUNK = 3 THEN 2260
[AH9] 2190 IF FUNK = 2 THEN LOCATE Y,1:PRINT L$;:LET Z
= Z - 1:GOTO 2100
[HWA] 2200 IF FUNK = 1 THEN LET EM = 1:COLOR 8,7:LOCATE
Y,1:PRINT L$;:GOTO 2140
[XHB] 2210 IF EM = 1 THEN PRINT #2,TEXT$:COLOR 7,0:LOCA
TE Y,1:PRINT L$;:LOCATE Y,1
[4M6] 2220 IF EM = 1 THEN PRINT TEXT$;:LET Z = Z + 1
[X98] 2230 IF EM = 1 AND Z > 19 THEN LET Z = 20:LET MZ
= 1:GOTO 2280
[2P5] 2240 IF EM = 1 THEN LET EM = 0:GOTO 2120
[E63] 2250 LET ETXT$ = TEXT$
[XEB] 2260 COLOR 7,0:LOCATE Y,1:PRINT L$;:LOCATE Y,1:PR
INT ETXT$;:PRINT #2,ETXT$
[0G6] 2270 IF MZ = 1 AND LEN(ETXT$) < 80 THEN PRINT
[GP3] 2280 IF MZ = 1 THEN PRINT
[7V7] 2290 IF MZ = 1 AND EM = 1 AND LEN(TEXT$) < 80 THE
N PRINT
[898] 2300 IF MZ = 1 AND EM = 1 THEN LET EM = 0:GOTO 21
20 ELSE 2100

```



```

[8U4] 2310 CLOSE #1:GOSUB 11720:PRINT
[AN0] 2320 '
[EN0] 2330 '
[4P9] 2340 REM weitere Texteingabe nach Erreichen des D
ateiendes
[PN0] 2350 '
[XKA] 2360 COLOR 16,7:PRINT " ";CHR$(4);:COLOR 8,7:PRIN
T " Das Dateiende von ";
[UHA] 2370 COLOR 16,7:PRINT DATEINAMES$;:COLOR 8,7:PRINT
" ist erreicht! ";
[U6B] 2380 PRINT TAB(49) "Sie können weiter Text eingebe
n ";:COLOR 7,0:PRINT
[H5B] 2390 LET Y = CSRLIN:VIEW PRINT:GOSUB 10680:VIEW P
RINT 3 TO 23:LET NA = 1
[Z92] 2400 LOCATE Y,1
[0H9] 2410 LET NAZ = CSRLIN:IF NAZ = 24 THEN LET NAZ =
23:PRINT CHR$(0);
[Z99] 2420 GOSUB 9360:IF FUNK = 4 THEN LET NA = 0:GOSUB
2670:GOTO 2500
[U39] 2430 IF FUNK = 1 OR FUNK = 2 OR FUNK = 3 OR FUNK
= 5 THEN 2420
[9E8] 2440 LOCATE AKT,1:PRINT TEXT$;SPACE$(80 - LEN(TE
XT$));
[4A4] 2450 PRINT #2,TEXT$:GOTO 2410
[XN0] 2460 '
[1P0] 2470 '
[ARA] 2480 REM Konvertierungsanzeigen nach Beendigung d
es Editierens
[9P0] 2490 '
[WY8] 2500 CLOSE:KILL DATEINAME$:NAME "TIMOP.NED" AS DA
TEINAMES$
[CY9] 2510 GOSUB 11400:FOR A = 11 TO 16:LOCATE A,22:PRI
NT SPACE$(36):NEXT
[L95] 2520 GOSUB 11720:IF RST = 1 THEN 2560
[D89] 2530 LOCATE 12,29:PRINT MN$(21):LOCATE 13,31 - LE
N(DATEINAMES$)/2
[2BA] 2540 PRINT "Textdatei ";:COLOR 8,7:PRINT " ";DATE
INAMES$; " ";:COLOR 7,0
[BM8] 2550 PRINT " ist":LOCATE 14,35:PRINT "beendet!":G
OTO 10280
[M89] 2560 LOCATE 11,29:PRINT MN$(21):LOCATE 12,31 - LE
N(DATEINAMES$)/2
[EBA] 2570 PRINT "Textdatei ";:COLOR 8,7:PRINT " ";DATE
INAMES$; " ";:COLOR 7,0
[U37] 2580 PRINT " ist":LOCATE 13,35:PRINT "beendet!"
[EP9] 2590 LOCATE 14,23:PRINT "Der abgetrennte Textteil
wurde in"
[NYB] 2600 LOCATE 15,23:PRINT "der Datei ";:COLOR 8,7:P
RINT " RÖDEL.NED ";:COLOR 7,0
[5V5] 2610 PRINT " gespeichert":LOCATE 16,23
[22B] 2620 PRINT "und kann weiter bearbeitet werden.":L
ET RST = 0:GOTO 10280
[RN0] 2630 '
[VN0] 2640 '
[3R9] 2650 REM kleinen Rahmen aufbauen mit der Warteauf
forderung
[3P0] 2660 '
[DK4] 2670 COLOR 7,0:VIEW PRINT:CLS
[BS9] 2680 LOCATE 8,20:COLOR 7,0:PRINT CHR$(201);STRING
$(38,205);CHR$(187)
[3P7] 2690 FOR A = 9 TO 16:LOCATE A,20:PRINT CHR$(186):
NEXT
[ZM8] 2700 LOCATE 17,20:PRINT CHR$(200);STRING$(38,205)
;CHR$(188)
[GY8] 2710 FOR A = 16 TO 9 STEP -1:LOCATE A,59:PRINT CH
R$(186):NEXT
[CP8] 2720 LOCATE 10,20:PRINT CHR$(199);STRING$(38,196)
;CHR$(182)
[D6A] 2730 LOCATE 9,22:COLOR 10,0:PRINT CHR$(3);TAB(31)
MN$(19);TAB(57) CHR$(3)
[9U5] 2740 COLOR 7,0:LOCATE 12,29:PRINT MN$(21)
[PM8] 2750 LOCATE 13,30 - LEN(DATEINAMES$)/2:PRINT "Text
datei ";
[K3A] 2760 COLOR 16,7:PRINT " ";DATEINAMES$; " ";:COLOR 7
,0:PRINT " läuft!"
[YV8] 2770 COLOR 25,0:LOCATE 15,31:PRINT MN$(22):COLOR
7,8:RETURN
[EP0] 2780 '
[KP0] 2790 '
[KV4] 2800 REM Restdatei abspeichern
[NN0] 2810 '
[SLA] 2820 OPEN "O",#2,"RÖDEL.NED":PRINT #2,ETXT$:GOSUB
11300:CLOSE:GOTO 2500
[XN0] 2830 '
[1P0] 2840 '
[MW3] 2850 REM Datei umbenennen
[9P0] 2860 '
[8SA] 2870 COLOR 8,7:GOSUB 7500:LET AM = 7:GOSUB 7820:O
N R GOTO 3090,1990,6150,6520
[E7A] 2880 IF AS = "A" OR AS = "a" THEN LET ALTERNAMES$
= DATEINAMES$:GOTO 2940

```

```

[MNA] 2890 IF AS = "R" OR AS = "r" THEN LET ALTERNAMES$
= "RÖDEL.NED":GOTO 2940
[9DA] 2900 IF AS = "S" OR AS = "s" THEN LET ALTERNAMES$
= "GASED.NED":GOTO 2940
[HFA] 2910 IF AS = "T" OR AS = "t" THEN LET ALTERNAMES$
= "TIMOP.NED":GOTO 2940
[KF6] 2920 IF AS = CHR$(13) THEN GOSUB 8510 ELSE 2870
[9ZB] 2930 IF EINGABES$ <> "" THEN LET ALTERNAMES$ = EING
ABES$ ELSE GOSUB 10200:GOTO 2870
[N4A] 2940 GOSUB 10200:LOCATE 14,45:COLOR 1,0:PRINT "Ge
ben Sie den neuen"
[DK9] 2950 LOCATE 15,45:PRINT "Dateinamen ein":COLOR 7,
0:GOSUB 8540
[YZB] 2960 IF EINGABES$ <> "" THEN LET NEUERNAMES$ = EING
ABES$ ELSE GOSUB 10200:GOTO 2870
[W0C] 2970 NAME ALTERNAMES$ AS NEUERNAMES$:GOSUB 10200:LO
CATE 14,45:PRINT "Die Datei ";
[BDB] 2980 COLOR 8,7:PRINT " ";ALTERNAMES$; " ";:COLOR 7,
0:PRINT " wurde":LOCATE 15,45
[VS8] 2990 PRINT "umbenannt. Sie existiert jetzt":LOCAT
E 16,45
[TFB] 3000 PRINT "unter dem Namen ";:COLOR 8,7:PRINT "
";NEUERNAMES$; " ";:COLOR 7,0
[1T9] 3010 LOCATE 17,45:PRINT "im aktuellen Verzeichnis
.":TIMER ON
[LF5] 3020 LET AS = INKEY$:IF AS = "" THEN 3020
[AR7] 3030 GOSUB 10200:GOSUB 7840:ON R GOTO 3090,1990,6
150,6520
[RW1] 3040 GOTO 2870
[FN0] 3050 '
[LN0] 3060 '
[VZ3] 3070 REM Dateien kopieren
[VN0] 3080 '
[6SA] 3090 COLOR 8,7:GOSUB 7510:LET AM = 8:GOSUB 7820:O
N R GOTO 3280,2870,6520,6870
[A5A] 3100 IF AS = "A" OR AS = "a" THEN LET KOPIENAMES$
= DATEINAMES$:GOTO 3150
[M9A] 3110 IF AS = "N" OR AS = "n" THEN LET KOPIENAMES$
= NEUERNAMES$:GOTO 3150
[PF9] 3120 IF AS = "Z" OR AS = "z" THEN LET KOPIENAMES$
= ZIEL$:GOTO 3150
[RE6] 3130 IF AS = CHR$(13) THEN GOSUB 8510 ELSE 3090
[0YB] 3140 IF EINGABES$ <> "" THEN LET KOPIENAMES$ = EING
ABES$ ELSE GOSUB 10200:GOTO 3090
[852] 3150 GOSUB 10200
[TSA] 3160 IF INSTR(1,KOPIENAMES$,"?") > 0 OR INSTR(1,KO
PIENAMES$,"*") > 0 THEN 3180
[VT8] 3170 LOCATE 14,45:PRINT "Wohin soll die Datei ":G
OTO 3190
[XY7] 3180 LOCATE 14,45:PRINT "Wohin sollen die Dateien
"
[LF9] 3190 LOCATE 15,45:COLOR 8,7:PRINT " ";KOPIENAMES$;
" ";:COLOR 7,0
[P96] 3200 PRINT "kopiert werden?":GOSUB 8600
[189] 3210 IF EINGABES$ = "" THEN GOSUB 10200:GOTO 3090
ELSE LET RM = 4
[KG7] 3220 LET COPY$ = "COPY " + KOPIENAMES$ + " " + PFA
D$ + "/"V"
[HR7] 3230 LOCATE 6,45:GOSUB 11610:SHELL COPY$:GOTO 102
80
[HN0] 3240 '
[NN0] 3250 '
[134] 3260 REM Dateien löschen
[XN0] 3270 '
[ENA] 3280 COLOR 8,7:GOSUB 7520:LET AM = 9:GOSUB 7820:O
N R GOTO 3550,3090,6870,850
[LF6] 3290 IF AS = CHR$(13) THEN GOSUB 8510 ELSE 3280
[6YB] 3300 IF EINGABES$ <> "" THEN LET LSCHNAME$ = EINGA
BES$ ELSE GOSUB 10200:GOTO 3280
[LE7] 3310 GOSUB 10200:LOCATE 14,45:PRINT "Sie wollen"
[WBA] 3320 IF INSTR(1,LSCHNAME$,"?") > 0 OR INSTR(1,LSCH
NAME$,"*") > 0 THEN 3340
[B27] 3330 LOCATE 15,45:PRINT "die Datei ";:GOTO 3350
[506] 3340 LOCATE 15,45:PRINT "die Dateien ";
[94A] 3350 COLOR 8,7:PRINT " ";LSCHNAME$; " ";:COLOR 7,0
:PRINT " löschen:"
[UGA] 3360 LOCATE 17,45:COLOR 1,0:PRINT "Sind Sie sich
sicher?":COLOR 7,0
[GRA] 3370 LOCATE 19,45:PRINT "Ja = > J <      Nein = >
N <":GOSUB 10210:TIMER ON
[6H5] 3380 LET AS = INKEY$:IF AS = "" THEN 3380
[3X8] 3390 TIMER OFF:IF AS = "N" OR AS = "n" THEN GOSUB
10200:GOTO 3280
[MW9] 3400 IF AS = "J" OR AS = "j" THEN KILL LSCHNAME$:
GOSUB 10200:GOTO 3430
[NP3] 3410 IF TON = 1 THEN BEEP
[WW1] 3420 GOTO 3380
[8CA] 3430 IF INSTR(1,LSCHNAME$,"?") > 0 OR INSTR(1,LSCH
NAME$,"*") > 0 THEN 3460
[PLA] 3440 LOCATE 14,45:PRINT "Die Datei ";:COLOR 9,0:P
RINT " ";LSCHNAME$; " "

```

Das Basic-Listing »bukamav.asc« ist nicht nur ein Texteditor


```

[CC9] 3450 LOCATE 15,45:COLOR 7,0:PRINT "wurde gelöscht
!":GOTO 3480
[2ZA] 3460 LOCATE 14,45:PRINT "Die Dateien ";:COLOR 9,0
:PRINT " ";:LSCNAMES$;" "
[5P9] 3470 LOCATE 15,45:COLOR 7,0:PRINT "wurden gelöscht
t!":TIMER ON
[DH5] 3480 LET AS = INKEY$:IF AS = "" THEN 3480
[WN7] 3490 GOSUB 10200:GOSUB 7840:ON R GOTO 3550,3090,6
870,850
[PW1] 3500 GOTO 3280
[EN0] 3510 '
[KN0] 3520 '
[NP4] 3530 REM Inhaltsverzeichnis
[UN0] 3540 '
[4YA] 3550 COLOR 8,7:GOSUB 7530:LET AM = 10:GOSUB 7820:
ON R GOTO 3860,3280,850,1200
[1L6] 3560 IF AS = CHR$(13) THEN GOSUB 7380:GOTO 3640
[VA2] 3570 LOCATE 6,45
[58A] 3580 IF AS = "A" OR AS = "a" THEN GOSUB 11610:SHE
LL "DIR A:/P":GOTO 3690
[M8A] 3590 IF AS = "B" OR AS = "b" THEN GOSUB 11610:SHE
LL "DIR B:/P":GOTO 3690
[Y7A] 3600 IF AS = "C" OR AS = "c" THEN GOSUB 11610:SHE
LL "DIR C:/P":GOTO 3690
[D8A] 3610 IF AS = "D" OR AS = "d" THEN GOSUB 11610:SHE
LL "DIR D:/P":GOTO 3690
[MKA] 3620 IF AS = "S" OR AS = "s" OR AS = "E" OR AS =
"e" THEN GOSUB 7380:GOTO 3700
[2X1] 3630 GOTO 3550
[Y7A] 3640 COLOR 1,0:LOCATE 14,45:PRINT "Geben Sie Lauf
werk/Verzeichnis"
[VRA] 3650 LOCATE 15,45:PRINT "bzw. Dateikennung ein:":
COLOR 7,0:GOSUB 8620
[YM9] 3660 LOCATE 6,45:IF ES = CHR$(0) + CHR$(15) THEN
GOSUB 10200:GOTO 3550
[A29] 3670 IF EINGABES$ = "" THEN GOSUB 11610:SHELL "DIR
/P":GOTO 3690
[0K8] 3680 LET VDIR$ = "DIR " + PFAD$ + "/P":GOSUB 1161
0:SHELL VDIR$
[TR3] 3690 LET RM = 5:GOTO 10280
[AUA] 3700 LOCATE 14,45:PRINT "Wohin soll die Protokoll
datei":LOCATE 15,45
[KCA] 3710 PRINT "geschrieben werden:":LET MAZ = 1:GOSU
B 8600:GOSUB 10200
[879] 3720 IF ES = "#" OR ES = "*" THEN LET PFAD$ = "":
GOSUB 10200:GOTO 3740
[YY6] 3730 IF EINGABES$ = "" THEN GOSUB 10200:GOTO 3550
[3LA] 3740 LOCATE 14,45:PRINT "Das Inhaltsverzeichnis v
om":LOCATE 15,45
[RW8] 3750 PRINT "aktuellen Laufwerk oder Unter-":LOCAT
E 16,45
[G9A] 3760 PRINT "verzeichnis wird in der Datei":LOCATE
17,45:COLOR 8,7
[LYA] 3770 PRINT "GASED.NED ";:COLOR 7,0:PRINT " proto
kolliert.":LOCATE 19,53
[SKA] 3780 COLOR 18,7:PRINT "> Bitte warten <":COLOR 7,
0:LET MAZ = 0:LET RM = 5
[6B7] 3790 LET PROTOKOLL$ = "DIR > " + PFAD$ + "GASED.N
ED"
[AY5] 3800 LET PROTDATIS$ = PFAD$ + "GASED.NED"
[G3A] 3810 LOCATE 6,45:SHELL PROTOKOLL$:OPEN "I",#1,PRO
TDATEIS$:GOTO 970
[VN0] 3820 '
[ZN0] 3830 '
[ZE6] 3840 REM Laufwerk/Verzeichnis wechseln
[7P0] 3850 '
[Y1B] 3860 COLOR 8,7:GOSUB 7540:LET AM = 11:GOSUB 7820:
ON R GOTO 4090,3550,1200,1360
[N88] 3870 IF AS = CHR$(13) THEN GOSUB 7380:GOSUB 8580:
GOTO 4020
[8B2] 3880 LOCATE 6,45
[1T9] 3890 IF AS = "A" OR AS = "a" THEN SHELL "A:":LET
MLWS$ = "A":GOTO 3990
[ES9] 3900 IF AS = "B" OR AS = "b" THEN SHELL "B:":LET
MLWS$ = "B":GOTO 3990
[0T9] 3910 IF AS = "C" OR AS = "c" THEN SHELL "C:":LET
MLWS$ = "C":GOTO 3990
[KT9] 3920 IF AS = "D" OR AS = "d" THEN SHELL "D:":LET
MLWS$ = "D":GOTO 3990
[4U9] 3930 IF AS = "E" OR AS = "e" THEN SHELL "E:":LET
MLWS$ = "E":GOTO 3990
[PU9] 3940 IF AS = "F" OR AS = "f" THEN SHELL "F:":LET
MLWS$ = "F":GOTO 3990
[8V9] 3950 IF AS = "G" OR AS = "g" THEN SHELL "G:":LET
MLWS$ = "G":GOTO 3990
[UV9] 3960 IF AS = "H" OR AS = "h" THEN SHELL "H:":LET
MLWS$ = "H":GOTO 3990
[CW9] 3970 IF AS = "I" OR AS = "i" THEN SHELL "I:":LET
MLWS$ = "I":GOTO 3990
[4Y1] 3980 GOTO 3860

```

```

[0X8] 3990 GOSUB 7750:GOSUB 12080:LOCATE 23,45:COLOR 10
,0:PRINT LW$
[358] 4000 IF AZG = 1 AND MAZG = 0 THEN GOSUB 7000:GOSU
B 7750
[FW1] 4010 GOTO 3860
[0B4] 4020 IF EINGABES$ = "" THEN 4040
[B1A] 4030 CHDIR PFAD$:IF AZG = 1 AND MAZG = 0 THEN GOS
UB 7000:GOSUB 7750
[UL3] 4040 GOSUB 10200:GOTO 3860
[HN0] 4050 '
[NN0] 4060 '
[Y36] 4070 REM Unterverzeichnis erstellen
[XN0] 4080 '
[82B] 4090 COLOR 8,7:GOSUB 7550:LET AM = 12:GOSUB 7820:
ON R GOTO 4420,3860,1360,1490
[448] 4100 IF AS <> CHR$(13) THEN 4090 ELSE TIMER ON:GO
SUB 7380
[9N8] 4110 LOCATE 14,45:PRINT "Unterverz. anlegen      =
> 1 <"
[YX8] 4120 LOCATE 15,45:PRINT "Unterverz. umbenennen   =
> 2 <"
[CY8] 4130 LOCATE 16,45:PRINT "Unterverz. löschen      =
> 3 <"
[DAA] 4140 LOCATE 17,45:PRINT "Unterverz. anzeigen     =
> 4 <":GOSUB 7620
[EG5] 4150 LET AS = INKEY$:IF AS = "" THEN 4150
[593] 4160 IF AS = "0" THEN 4360
[TC5] 4170 ON VAL(AS) GOTO 4200,4230,4310,4340
[AR3] 4180 IF TON = 1 THEN BEEP
[DX1] 4190 GOTO 4150
[AE7] 4200 GOSUB 10200:GOSUB 8580:IF EINGABES$ = "" THEN
4360
[D7A] 4210 MKDIR PFAD$:GOSUB 4370:LOCATE 14,45:PRINT "E
s wurde ein neues"
[20A] 4220 LOCATE 15,45:PRINT "Unterverzeichnis angeleg
t.":GOTO 4350
[PE7] 4230 GOSUB 10200:GOSUB 8580:IF EINGABES$ = "" THEN
4360
[134] 4240 LET ALTERNAMES$ = PFAD$
[G1C] 4250 LOCATE 14,45:COLOR 1,0:PRINT "Geben Sie Bitt
e den neuen Namen":LOCATE 15,45
[SL8] 4260 PRINT "des Unterverzeichnisses ein:":COLOR 7
,0
[0S5] 4270 GOSUB 8650:IF EINGABES$ = "" THEN 4360
[CF9] 4280 LET NEUERNAMES$ = "MOVE " + ALTERNAMES$ + " "
+ PFAD$:GOSUB 10210
[62C] 4290 SHELL NEUERNAMES$:GOSUB 4370:LOCATE 14,45:PRI
NT "Das Unterverzeichnis"
[248] 4300 LOCATE 15,45:PRINT "wurde umbenannt.":GOTO 4
350
[HE7] 4310 GOSUB 10200:GOSUB 8580:IF EINGABES$ = "" THEN
4360
[8XA] 4320 RMDIR PFAD$:GOSUB 4370:LOCATE 14,45:PRINT "D
as Unterverzeichnis"
[6A8] 4330 LOCATE 15,45:PRINT "wurde gelöscht.":GOTO 43
50
[HZ8] 4340 GOSUB 11610:SHELL "TREE/F|MORE":LET RM = 6:G
OTO 10280
[WG5] 4350 LET AS = INKEY$:IF AS = "" THEN 4350
[DM3] 4360 GOSUB 10200:GOTO 4090
[BF9] 4370 FOR A = 14 TO 16:LOCATE A,45:PRINT SPACE$(34
):NEXT:RETURN
[6P0] 4380 '
[AP0] 4390 '
[RX4] 4400 REM Disketten formatieren
[DN0] 4410 '
[52B] 4420 COLOR 8,7:GOSUB 7560:LET AM = 13:GOSUB 7820:
ON R GOTO 4860,4090,1490,1740
[BC8] 4430 IF AS <> CHR$(13) THEN 4420 ELSE LET SYS = 0
:GOSUB 7380
[U88] 4440 LOCATE 14,45:PRINT "180 KByte Laufwerk      =
> 1 <"
[098] 4450 LOCATE 15,45:PRINT "360 KByte Laufwerk      =
> 2 <"
[288] 4460 LOCATE 16,45:PRINT "1,2 MByte Laufwerk      =
> 3 <"
[HMA] 4470 LOCATE 17,45:PRINT "Diskettenamen ändern   =
> 4 <":GOSUB 7620
[CX1] 4480 TIMER ON
[RH5] 4490 LET AS = INKEY$:IF AS = "" THEN 4490
[LE7] 4500 TIMER OFF:IF AS = "0" THEN GOSUB 10190:GOTO
4420
[KB9] 4510 IF AS = "S" OR AS = "s" THEN GOSUB 4730:LET
SYS = 1:GOTO 4480
[AE5] 4520 ON VAL(AS) GOTO 4550,4570,4590,4620
[2R3] 4530 IF TON = 1 THEN BEEP
[GX1] 4540 GOTO 4490
[5W9] 4550 IF SYS = 1 THEN GOSUB 11610:SHELL "FORMAT A:
/1/S/V":GOTO 4610
[K17] 4560 GOSUB 11610:SHELL "FORMAT A:/1/V":GOTO 4610

```



```

[TW9] 4570 IF SYS = 1 THEN GOSUB 11610:SHELL "FORMAT A:
/4/S/V":GOTO 4610
[627] 4580 GOSUB 11610:SHELL "FORMAT A:/4/V":GOTO 4610
[PP9] 4590 IF SYS = 1 THEN GOSUB 11610:SHELL "FORMAT A:
/S/V":GOTO 4610
[FB5] 4600 GOSUB 11610:SHELL "FORMAT A:/V"
[DF3] 4610 LET RM = 7:GOTO 5000
[NVA] 4620 GOSUB 10190:LOCATE 14,45:COLOR 1,0:PRINT "Ge
ben Sie nun den neuen"
[2SA] 4630 LOCATE 15,45:PRINT "Diskettenamen ein:":COL
OR 7,0:LOCATE 17,45
[CZ8] 4640 PRINT "Der Name darf maximal 11 Zeichen":LOC
ATE 18,45
[R29] 4650 PRINT "lang sein. Zum Löschen des Namens":LO
CATE 19,45
[PTB] 4660 PRINT "geben Sie nur ":COLOR 8,7:PRINT " RE
TURN ":COLOR 7,0:PRINT " ein."
[ANA] 4670 LOCATE 6,45:PRINT "Diese Zeichen sind nicht
erlaubt:":COLOR 8,7
[GN8] 4680 LOCATE 8,45:PRINT " * ? / \ | . , ; : + = <
> [ ] ":COLOR 7,0
[N2B] 4690 LET X = 23:LET Y = 60:LET LN = 11:LOCATE X,Y
+ 1:PRINT STRING$(LN,219)
[A6A] 4700 GOSUB 9070:GOSUB 10210:IF EINGABES$ = "*" THE
N GOSUB 10190:GOTO 4420
[BC4] 4710 IF EINGABES$ = "" THEN 4780
[L3A] 4720 LET DISKNAMES$ = "LABEL A:" + EINGABES$:SHELL
DISKNAMES$:GOTO 4810
[N98] 4730 LOCATE 6,45:PRINT "Wenn Sie jetzt die Funkti
on"
[ELB] 4740 LOCATE 7,45:PRINT "> 1 <, > 2 < oder > 3 < w
ählen,":LOCATE 7,47:COLOR 18,7
[RWB] 4750 PRINT "1":LOCATE 7,54:PRINT "2":LOCATE 7,65:
PRINT "3":COLOR 7,0:LOCATE 8,45
[ORB] 4760 PRINT "werden die ":COLOR 1,0:PRINT "MS-DOS
- Systemdateien":COLOR 7,0
[08A] 4770 LOCATE 9,45:PRINT "auf die Diskette übertrag
en.":RETURN
[1Y6] 4780 LET LSCHLABEL$ = "LABEL A: > NUL < A:0.NED"
[5FA] 4790 OPEN "O",#1,"A:0.NED":PRINT #1,CHR$(13);"Y";
CHR$(13);"J";CHR$(13):CLOSE
[DD5] 4800 SHELL LSCHLABEL$:KILL "A:0.NED"
[995] 4810 GOSUB 7750:GOSUB 10190:GOTO 4420
[XN0] 4820 '
[1P0] 4830 '
[K54] 4840 REM Textdatei drucken
[9P0] 4850 '
[F2B] 4860 COLOR 8,7:GOSUB 7570:LET AM = 14:GOSUB 7820:
ON R GOTO 5310,4420,1740,1990
[87A] 4870 IF AS$ = "A" OR AS$ = "a" THEN LET DRUCKNAMES$
= DATEINAMES$:GOTO 4960
[GX4] 4880 IF AS$ = "D" OR AS$ = "d" THEN 4960
[PBA] 4890 IF AS$ = "N" OR AS$ = "n" THEN LET DRUCKNAMES$
= NEUERNAMES$:GOTO 4960
[NMA] 4900 IF AS$ = "R" OR AS$ = "r" THEN LET DRUCKNAMES$
= "RÖDEL.NED":GOTO 4960
[FDA] 4910 IF AS$ = "S" OR AS$ = "s" THEN LET DRUCKNAMES$
= "GASED.NED":GOTO 4960
[RFA] 4920 IF AS$ = "T" OR AS$ = "t" THEN LET DRUCKNAMES$
= "TIMOP.NED":GOTO 4960
[7G9] 4930 IF AS$ = "Z" OR AS$ = "z" THEN LET DRUCKNAMES$
= ZIEL$:GOTO 4960
[NR7] 4940 IF AS$ = CHR$(13) THEN LET M = 0:GOSUB 8510 E
LSE 4860
[10C] 4950 IF EINGABES$ <> "" THEN LET DRUCKNAMES$ = EING
ABES$ ELSE GOSUB 10200:GOTO 4860
[CAA] 4960 GOSUB 7380:OPEN "I",#1,DRUCKNAMES$:CLOSE:GOSU
B 10200:LOCATE 14,45
[X0C] 4970 PRINT "Sie haben die Datei ":COLOR 8,7:PRIN
T " ";DRUCKNAMES$;" ":COLOR 7,0
[GE9] 4980 LOCATE 15,45:PRINT "zum Drucken bestimmt:":L
OCATE 16,45
[WNA] 4990 PRINT "Seitenvorschub ein = ":COLOR 8,7:PR
INT MNS$(26):COLOR 7,0
[W5C] 5000 LOCATE 17,45:PRINT "Seitenvorschub aus =
> TAB <":GOSUB 7620:TIMER ON
[NY8] 5010 LET AS$ = INKEY$:IF AS$ = CHR$(13) THEN LET MD
R = 0:GOTO 5050
[6L6] 5020 IF AS$ = CHR$(9) THEN LET MDR = 1:GOTO 5050
[FY5] 5030 IF AS$ = "0" THEN GOSUB 10200:GOTO 4860
[LW1] 5040 GOTO 5010
[GG5] 5050 TIMER OFF:OPEN "I",#1,DRUCKNAMES$
[997] 5060 GOSUB 10190:LOCATE 14,45:PRINT "Die Datei ";
[LA9] 5070 COLOR 16,7:PRINT " ";DRUCKNAMES$;" ":COLOR 7,
0:LOCATE 15,45
[27A] 5080 PRINT "wird gedruckt.":TIMER ON:LET Z = 0:LE
T DM = 0:LET LZ = 0
[SV3] 5090 IF EOF(1) = -1 THEN 5160
[6R9] 5100 LET AS$ = INKEY$:IF AS$ = "Q" OR AS$ = "q" THEN
LET DM = 1:GOTO 5160

```

```

[CL5] 5110 LINE INPUT #1,TEXT$:LET Z = Z + 1
[NC9] 5120 IF LEN(TEXT$) < 80 THEN LPRINT TEXT$ ELSE LP
RINT TEXT$;
[KF3] 5130 IF MDR = 1 THEN 5090
[193] 5140 IF Z < 68 THEN 5090
[0N7] 5150 FOR A = 1 TO 4:LPRINT:NEXT:LET Z = 0:GOTO 50
90
[MR4] 5160 CLOSE #1:IF MDR = 1 THEN 5190
[K63] 5170 IF Z = 0 THEN 5190
[057] 5180 LET LZ = 72 - Z:FOR A = 1 TO LZ:LPRINT:NEXT
[0RA] 5190 TIMER OFF:LOCATE 14,55:COLOR 8,7:PRINT " ";D
RUCKNAMES$;" ":COLOR 7,8
[K9B] 5200 IF DM = 1 THEN LOCATE 15,45:PRINT "wurde zum
Teil gedruckt":GOTO 5220
[DF6] 5210 LOCATE 15,45:PRINT "wurde gedruckt."
[VN7] 5220 LOCATE 17,45:PRINT "Wiederholung = RETURN"
[XL5] 5230 LOCATE 18,53:PRINT "Ende = > Q <"
[VA5] 5240 TIMER ON:LET SPEICHER = FRE("")
[KG6] 5250 LET AS$ = INKEY$:IF AS$ = CHR$(13) THEN 5050
[C09] 5260 IF AS$ = "Q" OR AS$ = "q" THEN GOSUB 10200:GOT
O 4860 ELSE 5250
[1P0] 5270 '
[5P0] 5280 '
[2U9] 5290 REM Funktionstastenneubelegung (Datei- und P
fadnamen)
[8N0] 5300 '
[23B] 5310 COLOR 8,7:GOSUB 7580:LET AM = 15:GOSUB 7820:
ON R GOTO 6150,4860,1990,2870
[G34] 5320 IF AS$ = CHR$(13) THEN 5580
[499] 5330 IF AS$ = CHR$(0) + CHR$(94) THEN GOSUB 8510:L
ET FS(1) = EINGABES$
[C99] 5340 IF AS$ = CHR$(0) + CHR$(95) THEN GOSUB 8510:L
ET FS(2) = EINGABES$
[M99] 5350 IF AS$ = CHR$(0) + CHR$(96) THEN GOSUB 8510:L
ET FS(3) = EINGABES$
[W99] 5360 IF AS$ = CHR$(0) + CHR$(97) THEN GOSUB 8510:L
ET FS(4) = EINGABES$
[4A9] 5370 IF AS$ = CHR$(0) + CHR$(98) THEN GOSUB 8510:L
ET FS(5) = EINGABES$
[CA9] 5380 IF AS$ = CHR$(0) + CHR$(99) THEN GOSUB 8510:L
ET FS(6) = EINGABES$
[5A9] 5390 IF AS$ = CHR$(0) + CHR$(100) THEN GOSUB 8510:
LET FS(7) = EINGABES$
[A99] 5400 IF AS$ = CHR$(0) + CHR$(101) THEN GOSUB 8510:
LET FS(8) = EINGABES$
[M99] 5410 IF AS$ = CHR$(0) + CHR$(102) THEN GOSUB 8510:
LET FS(9) = EINGABES$
[5E9] 5420 IF AS$ = CHR$(0) + CHR$(103) THEN GOSUB 8510:
LET FS(10) = EINGABES$
[FE9] 5430 IF AS$ = CHR$(0) + CHR$(104) THEN GOSUB 8600:
LET FS(11) = EINGABES$
[UE9] 5440 IF AS$ = CHR$(0) + CHR$(105) THEN GOSUB 8600:
LET FS(12) = EINGABES$
[5F9] 5450 IF AS$ = CHR$(0) + CHR$(106) THEN GOSUB 8600:
LET FS(13) = EINGABES$
[GF9] 5460 IF AS$ = CHR$(0) + CHR$(107) THEN GOSUB 8600:
LET FS(14) = EINGABES$
[VF9] 5470 IF AS$ = CHR$(0) + CHR$(108) THEN GOSUB 8600:
LET FS(15) = EINGABES$
[6G9] 5480 IF AS$ = CHR$(0) + CHR$(109) THEN GOSUB 8600:
LET FS(16) = EINGABES$
[CF9] 5490 IF AS$ = CHR$(0) + CHR$(110) THEN GOSUB 8600:
LET FS(17) = EINGABES$
[KE9] 5500 IF AS$ = CHR$(0) + CHR$(111) THEN GOSUB 8600:
LET FS(18) = EINGABES$
[XE9] 5510 IF AS$ = CHR$(0) + CHR$(112) THEN GOSUB 8600:
LET FS(19) = EINGABES$
[CE9] 5520 IF AS$ = CHR$(0) + CHR$(113) THEN GOSUB 8600:
LET FS(20) = EINGABES$
[0M3] 5530 GOSUB 10200:GOTO 5310
[YN0] 5540 '
[2P0] 5550 '
[NKB] 5560 REM Tabelle der ASCII - Codes und Funktionst
astenneubelegung F1 - F6
[AP0] 5570 '
[GR7] 5580 LET MFN = 1:LET XT = 24:LET YT = 70:GOSUB 81
70
[FY8] 5590 LOCATE 1,23:PRINT CHR$(203):LOCATE 2,23:PRIN
T CHR$(186)
[M7A] 5600 PRINT CHR$(204);STRING$(21,205);CHR$(202);ST
RING$(56,205);CHR$(185);
[CV8] 5610 LOCATE 2,3:COLOR 10,0:PRINT CHR$(3);TAB(7) "
B U K A M A V"
[M5C] 5620 LOCATE 2,26:PRINT "Tabelle der Sonderzeichen
und ihrer ASCII - Codes";
[TK5] 5630 PRINT TAB(78) CHR$(3);" ":COLOR 7,0
[7NA] 5640 LOCATE 16,1:GOSUB 8260:LOCATE 19,1:GOSUB 826
0:LOCATE 23,1:GOSUB 8260
[N42] 5650 GOSUB 8310

```

Das Basic-Listing »bukamav.asc« ist nicht nur ein Texteditor


```

[URC] 5660 LOCATE 17,3:PRINT "Sie können jetzt die Funk-
tionstasten Nr. 1 bis Nr. 6 ";
[EPB] 5670 PRINT "mit Sonderzeichen Ihrer":LOCATE 18,3:
PRINT "Wahl belegen: ";
[P5C] 5680 COLOR 10,0:PRINT "F1 ≈      F2 ≈      F3 ≈
      F4 ≈      F5 ≈      F6 ≈";
[VFD] 5690 COLOR 7,0:LOCATE 18,23:PRINT "█":LOCATE 18
,33:PRINT "█":LOCATE 18,43
[PGF] 5700 PRINT "█":LOCATE 18,53:PRINT "█":LOCATE
18,63:PRINT "█":LOCATE 18,73
[YLD] 5710 PRINT "█":LOCATE 20,3:PRINT "Geben Sie bit-
te in den Eingabefeldern die ";
[Y59] 5720 PRINT "entsprechenden Zeichencodes ein,":LOC
ATE 21,3
[L6D] 5730 PRINT "wobei Ihnen die obengezeigte Tabelle
als Hilfestellung dienen soll."
[2SB] 5740 LOCATE 22,3:PRINT "Die Funktionstasten Nr. 7
bis Nr. 10 sind für die ";
[3U6] 5750 PRINT "Editorfunktion reserviert."
[LP3] 5760 GOSUB 8400:GOSUB 6030
[578] 5770 LOCATE 4,1:LET B = 123:FOR X = 1 TO 12:LOCAT
E X + 3,2
[5KB] 5780 FOR A = B TO B + 10:COLOR 7,0:PRINT " ";COL
OR 8,7:PRINT RIGHT$(STR$(A),3);
[MT8] 5790 COLOR 7,0:PRINT " ";:COLOR 10,0:PRINT CHR$(A
);" ";:NEXT
[7V7] 5800 LET B = B + 11:NEXT:COLOR 7,0:LET LN = 3:LET
X = 18
[C47] 5810 LET A$ = INKEY$:GOSUB 8120:IF A$ = "" THEN 5
810
[DP7] 5820 IF A$ = "Q" OR A$ = "q" THEN GOSUB 6040:GOTO
10390
[X08] 5830 LET Y = 22:LOCATE X,23:PRINT "█":GOSUB 907
0
[5R7] 5840 IF E$ = "Q" OR E$ = "q" THEN GOSUB 6040:GOTO
10390
[Z6A] 5850 IF EINGABE = 0 THEN GOSUB 6030 ELSE LET F1 =
EINGABE:GOSUB 6030
[F18] 5860 LET Y = 32:LOCATE X,33:PRINT "█":GOSUB 907
0
[HR7] 5870 IF E$ = "Q" OR E$ = "q" THEN GOSUB 6040:GOTO
10390
[F7A] 5880 IF EINGABE = 0 THEN GOSUB 6030 ELSE LET F2 =
EINGABE:GOSUB 6030
[128] 5890 LET Y = 42:LOCATE X,43:PRINT "█":GOSUB 907
0
[SP7] 5900 IF E$ = "Q" OR E$ = "q" THEN GOSUB 6040:GOTO
10390
[U6A] 5910 IF EINGABE = 0 THEN GOSUB 6030 ELSE LET F3 =
EINGABE:GOSUB 6030
[E18] 5920 LET Y = 52:LOCATE X,53:PRINT "█":GOSUB 907
0
[4R7] 5930 IF E$ = "Q" OR E$ = "q" THEN GOSUB 6040:GOTO
10390
[A7A] 5940 IF EINGABE = 0 THEN GOSUB 6030 ELSE LET F4 =
EINGABE:GOSUB 6030
[028] 5950 LET Y = 62:LOCATE X,63:PRINT "█":GOSUB 907
0
[GR7] 5960 IF E$ = "Q" OR E$ = "q" THEN GOSUB 6040:GOTO
10390
[U7A] 5970 IF EINGABE = 0 THEN GOSUB 6030 ELSE LET F5 =
EINGABE:GOSUB 6030
[K28] 5980 LET Y = 72:LOCATE X,73:PRINT "█":GOSUB 907
0
[WR7] 5990 IF E$ = "Q" OR E$ = "q" THEN GOSUB 6040:GOTO
10390
[96A] 6000 IF EINGABE = 0 THEN GOSUB 6030 ELSE LET F6 =
EINGABE:GOSUB 6030
[X27] 6010 LET A$ = INKEY$:GOSUB 8110:IF A$ = "" THEN 6
010
[819] 6020 IF A$ = "Q" OR A$ = "q" THEN GOSUB 6040:GOTO
10390 ELSE 5830
[RF3] 6030 COLOR 8,7:GOTO 6050
[D76] 6040 LOCATE 24,3:PRINT "Menü";:COLOR 10,0
[HM5] 6050 LOCATE 18,23:PRINT " ";CHR$(F1);" "
[UM5] 6060 LOCATE 18,33:PRINT " ";CHR$(F2);" "
[3N5] 6070 LOCATE 18,43:PRINT " ";CHR$(F3);" "
[CN5] 6080 LOCATE 18,53:PRINT " ";CHR$(F4);" "
[NN5] 6090 LOCATE 18,63:PRINT " ";CHR$(F5);" "
[LC8] 6100 LOCATE 18,73:PRINT " ";CHR$(F6);" ";:COLOR 7,
0:RETURN
[8N0] 6110 '
[CN0] 6120 '
[YL6] 6130 REM Textdateigröße feststellen
[MN0] 6140 '
[C2B] 6150 COLOR 8,7:GOSUB 7590:LET AM = 16:GOSUB 7820:
ON R GOTO 6520,5310,2870,3090
[Y7A] 6160 IF A$ = "A" OR A$ = "a" THEN LET ZAHLDAT$ =
DATEINAME$:GOTO 6240
[8CA] 6170 IF A$ = "N" OR A$ = "n" THEN LET ZAHLDAT$ =
NEUERNAMES$:GOTO 6240

```

```

[5PA] 6180 IF A$ = "R" OR A$ = "r" THEN LET ZAHLDAT$ =
"RODEL.NED":GOTO 6240
[ZEA] 6190 IF A$ = "S" OR A$ = "s" THEN LET ZAHLDAT$ =
"GASED.NED":GOTO 6240
[2GA] 6200 IF A$ = "T" OR A$ = "t" THEN LET ZAHLDAT$ =
"TIMOP.NED":GOTO 6240
[UH9] 6210 IF A$ = "Z" OR A$ = "z" THEN LET ZAHLDAT$ =
"ZIEL$:GOTO 6240
[NE6] 6220 IF A$ = CHR$(13) THEN GOSUB 8510 ELSE 6150
[5XB] 6230 IF EINGABE$ <> "" THEN LET ZAHLDAT$ = EING
ABE$ ELSE GOSUB 10200:GOTO 6150
[PBA] 6240 OPEN "I",#1,ZAHLDAT$:GOSUB 10200:LET Z = 0
:LET SZ = 0:LET DSZ = 1
[R69] 6250 LOCATE 14,45:COLOR 7,0:PRINT "Die Textdatei
";:COLOR 8,7
[NE8] 6260 PRINT " ";ZAHLDAT$;" ";:COLOR 7,0:LET Y =
POS(0) + 1
[G78] 6270 PRINT " wird":LOCATE 16,45:PRINT "durchgezäh-
lt:"
[249] 6280 LOCATE 18,53:COLOR 18,7:PRINT MN$(22):COLOR
7,0:TIMER ON
[BZ2] 6290 WHILE NOT EOF(1)
[RU7] 6300 LINE INPUT #1,TEXT$:LET Z = Z + 1:LET SZ = S
Z + 1
[2Y6] 6310 IF SZ > 68 THEN LET DSZ = DSZ + 1:LET SZ = 0
[BW3] 6320 LOCATE 16,58:PRINT Z
[G91] 6330 WEND
[HFB] 6340 CLOSE:GOSUB 11720:TIMER OFF:FOR A = Y TO 79:
LOCATE 14,A:PRINT " ":NEXT
[GD2] 6350 LOCATE 16,45
[WLA] 6360 IF Z = 0 THEN PRINT "enthält keine Texteintr-
agung.":GOTO 6430
[P14] 6370 PRINT "besteht aus";
[BA7] 6380 IF Z = 1 THEN PRINT " einer Zeile.":GOTO 643
0
[S35] 6390 COLOR 10,0:PRINT Z;:COLOR 7,0
[B8A] 6400 PRINT "Zeilen und":LOCATE 18,44:COLOR 10,0:P
RINT DSZ;:COLOR 7,0
[D28] 6410 PRINT "Druckseite";:IF DSZ > 1 THEN PRINT "n
";
[024] 6420 PRINT ".":SPACES(12):GOTO 6440
[735] 6430 LOCATE 18,54:PRINT SPACES(16)
[8H5] 6440 LET A$ = INKEY$:IF A$ = "" THEN 6440
[ZM5] 6450 LET SPEICHER = FRE(""):TIMER OFF
[AR7] 6460 GOSUB 10200:GOSUB 7840:ON R GOTO 6520,5310,2
870,3090
[UX1] 6470 GOTO 6150
[DP0] 6480 '
[HP0] 6490 '
[HP7] 6500 REM Programmmerkürungen (Dokumentation)
[MN0] 6510 '
[F2B] 6520 COLOR 8,7:GOSUB 7600:LET AM = 17:GOSUB 7820:
ON R GOTO 6870,6150,3090,3280
[1D6] 6530 IF A$ <> CHR$(13) THEN 6520 ELSE LET MH = 1
[9KA] 6540 GOSUB 8170:LOCATE 3,1:GOSUB 8260:LOCATE 23,1
:GOSUB 8260:GOSUB 8310
[XEB] 6550 LOCATE 2,3:COLOR 10,0:PRINT CHR$(3);TAB(78)
CHR$(3):LOCATE 2,11:COLOR 9,0
[ZLA] 6560 PRINT " B U K A M A V - Dokumentation (Hilfs-
textdatei) Seite 1 "
[R7B] 6570 GOSUB 8400:LET MFN = 1:LET XT = 24:LET YT =
70:LET Z = 1:RESTORE 13150
[XG2] 6580 FOR A = 1 TO 19
[M4A] 6590 READ HTXT1$,HTXT2$:LOCATE A + 3,3:PRINT HT
XT1$;TAB(41) HTXT2$
[W91] 6600 NEXT
[ZC2] 6610 LOCATE 24,23
[S63] 6620 IF Z = 5 THEN 6650
[5HB] 6630 PRINT "Vor = > ";:COLOR 18,7:PRINT "Leertast
e";:COLOR 7,0:PRINT "< ";
[V63] 6640 IF Z = 1 THEN 6660
[XUA] 6650 PRINT "Zurück = > ";:COLOR 18,7:PRINT "TAB";
:COLOR 7,0:PRINT "< ";
[EX1] 6660 TIMER ON
[5K5] 6670 LET A$ = INKEY$:IF A$ = "" THEN 6670
[HC6] 6680 TIMER OFF:IF A$ = "Q" OR A$ = "q" THEN 6820
[327] 6690 IF A$ = " " OR A$ = CHR$(13) THEN LET Z = Z
+ 1:GOTO 6730
[HZ8] 6700 IF A$ = CHR$(9) OR A$ = CHR$(8) THEN LET Z =
Z - 1:GOTO 6720
[KX1] 6710 GOTO 6660
[385] 6720 IF Z < 1 THEN LET Z = 1:GOTO 6660
[V85] 6730 IF Z > 5 THEN LET Z = 5:GOTO 6660
[5U7] 6740 FOR A = 4 TO 22:LOCATE A,3:PRINT SPACES(76):
NEXT
[DR6] 6750 LOCATE 2,68:COLOR 9,0:PRINT Z:COLOR 7,0
[TW9] 6760 LOCATE 24,23:PRINT SPACES(40);:ON Z GOTO 677
0,6780,6790,6800,6810
[674] 6770 RESTORE 13150:GOTO 6580
[A74] 6780 RESTORE 13530:GOTO 6580
[E74] 6790 RESTORE 13910:GOTO 6580

```



```

[U64] 6800 RESTORE 14290:GOTO 6580
[Y64] 6810 RESTORE 14670:GOTO 6580
[FR7] 6820 LOCATE 24,3:PRINT "Menü";:LET MH = 0:GOTO 10
270
[5P0] 6830 '
[9P0] 6840 '
[3R9] 6850 REM BUKAMAV beenden und Rückkehr zum Betrieb
ssystem
[HP0] 6860 '
[LYA] 6870 COLOR 8,7:GOSUB 7610:LET AM = 18:GOSUB 7820:
ON R GOTO 850,6520,3280,3550
[Y54] 6880 IF AS <> CHR$(13) THEN 6870
[M2A] 6890 GOSUB 7380:TIMER ON:LOCATE 7,45:PRINT "Bitte
denken Sie an"
[E67] 6900 LOCATE 8,45:PRINT "Ihre Datensicherung."
[YY7] 6910 LOCATE 14,45:PRINT "Bitte bestätigen Sie mit
"
[TR7] 6920 LOCATE 16,51:COLOR 8,7:PRINT MN$(26):COLOR 7
,0
[8H8] 6930 LOCATE 18,45:PRINT MN$(28):LOCATE 19,45:PRIN
T MN$(29)
[US7] 6940 LET AS = INKEY$:IF AS = CHR$(13) THEN CLS:SY
STEM
[FEA] 6950 IF AS = "Q" OR AS = "q" OR AS = " " THEN GOS
UB 10190:GOTO 6870 ELSE 6940
[MP0] 6960 '
[SP0] 6970 '
[LL6] 6980 REM Aktuelles Verzeichnis einlesen
[OR0] 6990 '
[1T8] 7000 LOCATE 1,1,0:IF DOSALT = 1 THEN SHELL "CHDIR
":RETURN
[51A] 7010 LOCATE 2,8:PRINT SPACE$(66):LOCATE 2,8:SHELL
"CHDIR":RETURN
[BN0] 7020 '
[FN0] 7030 '
[4K6] 7040 REM Aktuelles Verzeichnis anzeigen
[RN0] 7050 '
[4D7] 7060 TIMER OFF:IF AZG = 0 THEN LOCATE 2,1:PRINT L
$
[NWA] 7070 IF AZG = 1 THEN LOCATE 2,1:PRINT L$:GOSUB 76
90:LET AZG = 0:RETURN
[304] 7080 IF DOSALT = 1 THEN 7100
[C0B] 7090 LOCATE 2,1:PRINT STRING$(5,176);CHR$(16);TAB
(75) CHR$(17);STRING$(5,176)
[9K6] 7100 GOSUB 7000:GOSUB 7750:LET AZG = 1:RETURN
[AN0] 7110 '
[EN0] 7120 '
[RE5] 7130 REM Aktuelle Dateien anzeigen
[PN0] 7140 '
[R64] 7150 GOSUB 7380:LET MDAZG = 1
[AH9] 7160 LOCATE 14,45:IF DATEINAME$ = "" THEN PRINT M
N$(31):GOTO 7180
[S8C] 7170 PRINT "Aktuelle Textdatei: ";:COLOR 8,7:PRIN
T " ";DATEINAME$;": "":COLOR 7,0
[GM6] 7180 LOCATE 15,45:IF DRUCKNAME$ = "" THEN 7210
[ZU8] 7190 PRINT "Zum Drucken bereitgestellt":LOCATE 1
6,45
[ZAB] 7200 PRINT "Datei: ";:COLOR 8,7:PRINT " ";DRUCKNA
ME$;": "":COLOR 7,0:GOTO 7230
[F37] 7210 PRINT "Es wurde keine Datei zum Drucken"
[HM6] 7220 LOCATE 16,45:PRINT "bereitgestellt."
[RB3] 7230 GOSUB 7280:RETURN
[TN0] 7240 '
[XN0] 7250 '
[BX5] 7260 REM Aktuellen Zielpfad anzeigen
[5P0] 7270 '
[9WB] 7280 LOCATE 17,45:IF PFAD$ = "" THEN PRINT "Kein
Zielpfad definiert.":RETURN
[9R6] 7290 COLOR 1,0:PRINT "Aktueller Zielpfad: "
[0A7] 7300 LOCATE 18,45:COLOR 8,7:PRINT LEFT$(PFAD$,33)
[XH4] 7310 IF LEN(PFAD$) < 34 THEN 7330
[ZT7] 7320 LOCATE 19,45:PRINT RIGHT$(PFAD$,LEN(PFAD$) -
33)
[963] 7330 COLOR 7,0:RETURN
[WN0] 7340 '
[0P0] 7350 '
[SU8] 7360 REM Aktuelle Datei- und Zielpfadsanzeige lös
chen
[8P0] 7370 '
[FY6] 7380 IF MDAZG = 1 THEN GOSUB 10200:LET MDAZG = 0
[HW1] 7390 RETURN
[FN0] 7400 '
[LN0] 7410 '
[N58] 7420 REM Menüpunkte definieren und positionieren
[VN0] 7430 '
[116] 7440 LOCATE 6,3,0:PRINT MN$(1):GOTO 7640
[916] 7450 LOCATE 7,3,0:PRINT MN$(2):GOTO 7640
[H16] 7460 LOCATE 8,3,0:PRINT MN$(3):GOTO 7640
[T16] 7470 LOCATE 9,3,0:PRINT MN$(4):GOTO 7640

```

```

[A16] 7480 LOCATE 10,3,0:PRINT MN$(5):GOTO 7640
[M16] 7490 LOCATE 11,3,0:PRINT MN$(6):GOTO 7640
[T06] 7500 LOCATE 12,3,0:PRINT MN$(7):GOTO 7640
[316] 7510 LOCATE 13,3,0:PRINT MN$(8):GOTO 7640
[D16] 7520 LOCATE 14,3,0:PRINT MN$(9):GOTO 7640
[X46] 7530 LOCATE 15,3,0:PRINT MN$(10):GOTO 7640
[856] 7540 LOCATE 16,3,0:PRINT MN$(11):GOTO 7640
[L56] 7550 LOCATE 17,3,0:PRINT MN$(12):GOTO 7640
[Y56] 7560 LOCATE 18,3,0:PRINT MN$(13):GOTO 7640
[966] 7570 LOCATE 19,3,0:PRINT MN$(14):GOTO 7640
[S56] 7580 LOCATE 20,3,0:PRINT MN$(15):GOTO 7640
[366] 7590 LOCATE 21,3,0:PRINT MN$(16):GOTO 7640
[956] 7600 LOCATE 22,3,0:PRINT MN$(17):GOTO 7640
[M56] 7610 LOCATE 23,3,0:PRINT MN$(18):GOTO 7640
[U47] 7620 LOCATE 18,45:PRINT "Ende
> 0 <"
[K16] 7630 LOCATE 19,45:COLOR 1,0:PRINT MN$(30)
[P63] 7640 COLOR 7,0:RETURN
[9P0] 7650 '
[DP0] 7660 '
[YU5] 7670 REM Menüüberschrift anzeigen
[NP0] 7680 '
[MY8] 7690 LOCATE 2,4:PRINT "****";TAB(75) "****":LOCATE
2,18:COLOR 1,0
[7MC] 7700 PRINT " " Textverarbeitung und Dateiverwal
tung " ":COLOR 7,0:RETURN
[WN0] 7710 '
[0P0] 7720 '
[216] 7730 REM Copyright - Vermerk anzeigen
[8P0] 7740 '
[AU3] 7750 LOCATE 25,3:COLOR 7,0
[EVA] 7760 PRINT "BUKAMAV - Version 1.1, ";CHR$(3);" 31
.1.1988, Wilhelm Andreas";
[NW1] 7770 RETURN
[SP0] 7780 '
[WP0] 7790 '
[XX3] 7800 REM Tastaturabfrage
[ZN0] 7810 '
[WU3] 7820 TIMER ON:LET MAZG = 0
[4K5] 7830 LET AS = INKEY$:IF AS = "" THEN 7830
[LA5] 7840 GOSUB 8060:TIMER OFF:LET RM = 0
[AR9] 7850 IF AS = CHR$(0) + CHR$(59) THEN GOSUB 7060:L
ET MAZG = 1:GOTO 7820
[VP7] 7860 IF AS = CHR$(0) + CHR$(60) THEN GOSUB 7150:G
OTO 7820
[ZU7] 7870 IF AS = CHR$(0) + CHR$(61) THEN GOSUB 10860:
GOTO 7820
[4U7] 7880 IF AS = CHR$(0) + CHR$(62) THEN GOSUB 11030:
GOTO 7820
[RM9] 7890 IF AS = CHR$(0) + CHR$(63) THEN LET TON = 1:
GOSUB 11660:GOTO 7820
[SL9] 7900 IF AS = CHR$(0) + CHR$(64) THEN LET TON = 0:
GOSUB 11660:GOTO 7820
[PE8] 7910 IF AS = CHR$(0) + CHR$(68) THEN LET MDAZG =
0:GOSUB 10200
[DH5] 7920 IF AS = " " THEN LET R = 1:GOTO 7990
[XP9] 7930 IF AS = CHR$(0) + CHR$(80) OR AS = CHR$(9) T
HEN LET R = 1:GOTO 7990
[5R9] 7940 IF AS = CHR$(0) + CHR$(72) OR AS = CHR$(8) T
HEN LET R = 2:GOTO 7990
[FK7] 7950 IF AS = CHR$(0) + CHR$(15) THEN LET R = 2:GO
TO 7990
[SK7] 7960 IF AS = CHR$(0) + CHR$(81) THEN LET R = 3:GO
TO 7990
[3L7] 7970 IF AS = CHR$(0) + CHR$(73) THEN LET R = 4:GO
TO 7990
[U33] 7980 LET R = 0:RETURN
[UUV] 7990 ON AM GOSUB 7440,7450,7460,7470,7480,7490,75
00,7510,7520,7530,7540,7550
[0Y6] 8000 IF AM > 12 THEN LET AM = AM - 12 ELSE RETURN
[DD7] 8010 ON AM GOSUB 7560,7570,7580,7590,7600,7610:RE
TURN
[DN0] 8020 '
[HN0] 8030 '
[MN5] 8040 REM Tastaturpuffer löschen
[TN0] 8050 '
[VE6] 8060 DEF SEG = 0:POKE 1050,PEEK(1052):RETURN
[1P0] 8070 '
[5P0] 8080 '
[T83] 8090 REM Zeitanzeige
[8N0] 8100 '
[SH5] 8110 LET MX = CSRLIN:LET MY = POS(0)
[UXB] 8120 LOCATE XT,YT,0:COLOR 8,7:PRINT TIME$;:COLOR
7,0:LOCATE MX,MY,0:RETURN
[MN0] 8130 '
[SN0] 8140 '
[ZW3] 8150 REM Rahmen aufbauen
[0P0] 8160 '
[BG7] 8170 CLS:PRINT CHR$(201);STRING$(78,205);CHR$(187
);

```

Das Basic-Listing »bukamav.asc« ist nicht nur ein Texteditor


```

[CK7] 8180 FOR A = 2 TO 24:LOCATE A,1:PRINT CHR$(186);:
NEXT
[YN8] 8190 LOCATE 25,1:PRINT CHR$(200);STRING$(78,205);
CHR$(188);
[V49] 8200 FOR A = 24 TO 2 STEP -1:LOCATE A,80:PRINT CH
R$(186);:NEXT
[GV1] 8210 RETURN
[LN0] 8220 '
[RN0] 8230 '
[ZS4] 8240 REM Querlinie darstellen
[ZN0] 8250 '
[P38] 8260 PRINT CHR$(199);STRING$(78,196);CHR$(182);:R
ETURN
[7P0] 8270 '
[BP0] 8280 '
[C59] 8290 REM kleine Felder innerhalb des Rahmens dars
tellen
[EN0] 8300 '
[V5B] 8310 LOCATE 23,16:PRINT CHR$(194);LOCATE 23,21:PR
INT CHR$(194);LOCATE 23,67
[1V2] 8320 PRINT CHR$(194)
[K1A] 8330 LOCATE 24,16:PRINT CHR$(179);TAB(21) CHR$(17
9);TAB(67) CHR$(179);
[MAB] 8340 LOCATE 25,16:PRINT CHR$(207);:LOCATE 25,21:P
RINT CHR$(207);:LOCATE 25,67
[EA4] 8350 PRINT CHR$(207);:RETURN
[6P0] 8360 '
[AP0] 8370 '
[AN7] 8380 REM Anzeigen innerhalb der kleinen Felder
[KP0] 8390 '
[HCA] 8400 LOCATE 24,3:COLOR 7,0:PRINT "Ende = > <";:
LOCATE 24,12:COLOR 18,7
[CNB] 8410 PRINT "Q";:LOCATE 24,18:COLOR 10,0:PRINT LWS
;:COLOR 7,0:IF MH = 1 THEN 8460
[B5B] 8420 IF DATEINAME$ = "" THEN LOCATE 24,23:PRINT M
N$(31);SPACE$(8);:GOTO 8460
[9E9] 8430 LOCATE 24,23:COLOR 8,7:PRINT " ";DATEINAME$;
";:COLOR 7,0
[DLB] 8440 PRINT "Lesen = > ";:COLOR 18,7:PRINT "L";:C
OLOR 7,0:PRINT "< Edit. = >";
[7L8] 8450 COLOR 18,7:PRINT "E";:COLOR 7,0:PRINT "<";
TAB(66) " ";
[LD7] 8460 LOCATE 24,69:PRINT STRING$(10,219);:RETURN
[DP0] 8470 '
[HP0] 8480 '
[2Y6] 8490 REM Eingabe des Datei- oder Pfadnamens
[MN0] 8500 '
[WG3] 8510 GOSUB 7380:COLOR 1,0
[P29] 8520 LOCATE 14,45:PRINT "Geben Sie den Dateinamen
oder den"
[7G8] 8530 LOCATE 15,45:PRINT "Programmnamen ein:":COLO
R 7,0
[4D9] 8540 LOCATE 17,45:PRINT "Wenn Sie keine Dateiname
nserwei-"
[G69] 8550 LOCATE 18,45:PRINT "terung angeben, wird die
Eingabe"
[Z28] 8560 LOCATE 19,45:PRINT "als BUKAMAV - Datei aufg
efasst."
[V55] 8570 LET M = 0:GOSUB 11450:GOTO 8660
[UBC] 8580 LOCATE 14,45:COLOR 1,0:PRINT "Geben Sie Bitt
e das gewünschte":LOCATE 15,45
[AL8] 8590 PRINT "Unterverzeichnis ein:":COLOR 7,0:GOTO
8620
[VG3] 8600 GOSUB 7380:COLOR 1,0
[BP9] 8610 LOCATE 16,45:PRINT "Geben Sie den Pfadnamen
ein:":COLOR 7,0
[1P9] 8620 LOCATE 17,45:PRINT "Pfad übernehmen/anzeigen
=> Tab <"
[M28] 8630 LOCATE 18,45:PRINT "Pfadname löschen => Alt
< + > Q <"
[A9A] 8640 LOCATE 19,45:PRINT "Funktion beenden => ";C
HR$(24);" < + > Tab <"
[F47] 8650 LET BUK = 0:LET MP = 0:LET M = 1:GOSUB 11500
[R97] 8660 LOCATE 23,61:COLOR 7,0:PRINT STRING$(12,219)
[YD9] 8670 LET EINGABES$ = "":FOR Z = 1 TO 12:LET AS(Z)
= "":NEXT:LET Z = 0
[EF6] 8680 LOCATE 23,60 + Z,0:COLOR 8,7:PRINT AS(Z)
[R49] 8690 IF Z < 12 THEN LOCATE 23,61 + Z,0:COLOR 16,7
:PRINT CHR$(4)
[D28] 8700 IF Z < 11 THEN LOCATE 23,62 + Z,0:COLOR 8,7:
PRINT " "
[T19] 8710 TIMER ON:LET Z = Z + 1:IF M = 0 AND Z < 10 T
HEN LET BUK = 1
[HK5] 8720 LET ES = INKEY$:IF ES = "" THEN 8720
[4X4] 8730 TIMER OFF:IF ES = " " THEN 8660
[C09] 8740 IF MAZ = 1 AND ES = "#" OR MAZ = 1 AND ES =
"" THEN RETURN
[WP8] 8750 IF ES = CHR$(0) + CHR$(15) THEN LET EINGABES
= "":GOTO 9020

```

```

[WD9] 8760 IF M = 1 AND ES = CHR$(0) + CHR$(16) THEN LE
T PFAD$ = "":GOTO 9000
[H95] 8770 IF M = 0 AND ES = CHR$(9) THEN 8720
[E0A] 8780 IF Z = 1 AND ES = CHR$(8) OR Z = 1 AND ES =
CHR$(0) + CHR$(75) THEN 8720
[WV8] 8790 IF Z = 1 AND ES = CHR$(9) AND M = 1 THEN LET
MP = 1:GOTO 9000
[U95] 8800 IF ES = CHR$(9) AND M = 1 THEN 8960
[CKA] 8810 IF Z = 1 AND ES = CHR$(13) AND MP = 1 THEN L
ET EINGABES$ = "":GOTO 9020
[NC5] 8820 IF Z = 1 AND ES = CHR$(13) THEN 9020
[C09] 8830 IF ES = "." AND M = 0 THEN LET BUK = 0:LOCAT
E 23,60 + Z,0:COLOR 8,7
[D09] 8840 IF ES = "." AND M = 0 THEN FOR Z = 10 TO 12:
LET AS(Z) = "":NEXT
[W87] 8850 IF ES = "." AND M = 0 AND POS(0) < 70 THEN P
RINT " "
[P2A] 8860 IF ES = "." AND M = 0 THEN LOCATE 23,72:PRIN
T " ":LET Z = 9:GOTO 8930
[RV8] 8870 IF ES = CHR$(8) OR ES = CHR$(0) + CHR$(75) T
HEN 8880 ELSE 8890
[Y66] 8880 LET AS(Z - 1) = "":LET Z = Z - 2:GOTO 8680
[B55] 8890 IF ES = CHR$(13) AND BUK = 1 THEN 8940
[D44] 8900 IF ES = CHR$(13) THEN 8960
[784] 8910 IF Z = 13 THEN LET Z = 12
[V47] 8920 IF Z = 9 AND M = 0 THEN LET ES = "":LET BUK
= 0
[735] 8930 LET AS(Z) = LEFT$(ES,1):GOTO 8680
[H48] 8940 FOR Z = 1 TO 8:LET EINGABES$ = EINGABES$ + AS(
Z):NEXT
[FV6] 8950 LET EINGABES$ = EINGABES$ + ".BUK":GOTO 9020
[YRA] 8960 FOR Z = 1 TO 12:LET EINGABES$ = EINGABES$ + AS
(Z):NEXT:IF M = 0 THEN 9020
[518] 8970 IF MP = 0 THEN LET PFAD$ = "":GOSUB 10220:LE
T MP = 1
[0M4] 8980 LET PFAD$ = PFAD$ + EINGABES$
[E27] 8990 IF LEN(PFAD$) > 63 THEN LET PFAD$ = LEFT$(PF
AD$,63)
[LVB] 9000 GOSUB 10220:LOCATE 17,54:COLOR 1,0:PRINT " P
fadname: ":COLOR 7,0:GOSUB 7300
[KB7] 9010 IF ES = CHR$(9) OR ES = CHR$(0) + CHR$(16) T
HEN 8660
[AR6] 9020 COLOR 7,0:GOSUB 10210:GOSUB 11400:RETURN
[LN0] 9030 '
[RN0] 9040 '
[RD8] 9050 REM Eingabe von ASCII - Codes und Diskettenn
amen
[ZN0] 9060 '
[SH9] 9070 LET EINGABES$ = "":FOR Z = 1 TO LN:LET AS(Z)
= "":NEXT:LET Z = 0
[3S6] 9080 LOCATE X,Y + Z,0:COLOR 8,7:PRINT AS(Z)
[Z89] 9090 IF Z < LN THEN LOCATE X,Y + 1 + Z,0:COLOR 16
,7:PRINT CHR$(4)
[GV8] 9100 IF Z < LN - 1 THEN LOCATE X,Y + 2 + Z,0:COLO
R 8,7:PRINT " "
[WA2] 9110 LET Z = Z + 1
[U39] 9120 LET ES = INKEY$:GOSUB 8120:IF ES = "" OR ES
= CHR$(9) THEN 9120
[DN8] 9130 IF ES = "Q" AND LN = 3 OR ES = "q" AND LN =
3 THEN RETURN
[B0A] 9140 IF Z = 1 AND ES = CHR$(0) + CHR$(15) THEN LE
T EINGABES$ = "":GOTO 9230
[XY9] 9150 IF Z = 1 AND ES = CHR$(8) OR Z = 1 AND ES =
CHR$(0) + CHR$(75) THEN 9120
[FW8] 9160 IF Z = 1 AND ES = CHR$(13) THEN LET EINGABES$
= "":GOTO 9230
[YT8] 9170 IF ES = CHR$(8) OR ES = CHR$(0) + CHR$(75) T
HEN 9180 ELSE 9190
[066] 9180 LET AS(Z - 1) = "":LET Z = Z - 2:GOTO 9080
[Z44] 9190 IF ES = CHR$(13) THEN 9220
[ZV4] 9200 IF Z = LN + 1 THEN LET Z = LN
[A25] 9210 LET AS(Z) = LEFT$(ES,1):GOTO 9080
[K88] 9220 FOR Z = 1 TO LN:LET EINGABES$ = EINGABES$ + AS
(Z):NEXT
[P26] 9230 LET EINGABES$ = VAL(EINGABES$):RETURN
[XN0] 9240 '
[1P0] 9250 '
[39A] 9260 REM Datei- und Laufwerksanzeige während der
Texteingabe
[9P0] 9270 '
[DKB] 9280 LOCATE 1,50:COLOR 7,0:PRINT CHR$(16);" ";:CO
LOR 10,0:PRINT LWS;:COLOR 7,0
[X97] 9290 PRINT " ";CHR$(17);:COLOR 8,7:PRINT " Datei:
";
[CF9] 9300 COLOR 7,0:PRINT CHR$(16);" ";:COLOR 10,0:PRI
NT DATEINAME$;
[DV9] 9310 COLOR 7,0:PRINT " ";CHR$(17);:COLOR 8,7:PRIN
T TAB(80) " ":RETURN
[SN0] 9320 '
[WN0] 9330 '

```



```

[KN4] 9340 REM Textzeilen eingeben
[4P0] 9350 '
[624] 9360 LET EINF = 0:COLOR 8,7
[BYB] 9370 IF NA = 1 AND ZUEN = 0 THEN LOCATE NAZ,1:COL
OR 8,7:PRINT LZ$;:LOCATE NAZ,1
[C18] 9380 IF EINF = 0 THEN LOCATE ,,1,0,12 ELSE LOCATE
,,1,6,12
[6P5] 9390 LET Z$ = INKEY$:IF Z$ = "" THEN 9390
[HS6] 9400 GOSUB 8060:LET AKTZ = CSRLIN:LET ZZ = 80
[KY5] 9410 IF Z$ = CHR$(9) AND EINF = 1 THEN 9390
[ZF9] 9420 IF Z$ = CHR$(9) AND POS(0) > 72 THEN LOCATE
AKTZ,80:GOTO 9390
[Z86] 9430 IF Z$ = CHR$(0) + CHR$(83) THEN LET Z$ = " "
[BC7] 9440 IF ZUEN = 0 AND Z$ = CHR$(0) + CHR$(81) THEN
9390
[018] 9450 IF POS(0) < 80 THEN LET ENDZ = 0 ELSE LET EN
DZ = 1
[388] 9460 IF Z$ = CHR$(0) + CHR$(92) THEN LET ZUEN = 1
:GOTO 9990
[424] 9470 IF Z$ = CHR$(8) THEN 9770
[TY9] 9480 IF Z$ = CHR$(0) + CHR$(82) AND EINF = 0 THEN
LET EINF = 1:GOTO 9380
[VY9] 9490 IF Z$ = CHR$(0) + CHR$(82) AND EINF = 1 THEN
LET EINF = 0:GOTO 9380
[G4B] 9500 IF Z$ = CHR$(0) + CHR$(77) AND POS(0) < 80
THEN PRINT CHR$(28);:GOTO 9390
[URA] 9510 IF Z$ = CHR$(0) + CHR$(75) AND POS(0) < 1 T
HEN PRINT CHR$(29);:GOTO 9390
[5NA] 9520 IF Z$ = CHR$(0) + CHR$(72) AND ZUEN = 1 THEN
PRINT CHR$(30);:GOTO 9390
[9NA] 9530 IF Z$ = CHR$(0) + CHR$(80) AND ZUEN = 1 THEN
PRINT CHR$(31);:GOTO 9390
[687] 9540 IF Z$ = CHR$(0) + CHR$(73) AND ZUEN = 0 THEN
9990
[LC7] 9550 IF Z$ = CHR$(0) + CHR$(73) AND ZUEN = 1 THEN
10010
[7L8] 9560 IF Z$ = CHR$(0) + CHR$(71) THEN LOCATE AKTZ,
1:GOTO 9390
[SN8] 9570 IF Z$ = CHR$(0) + CHR$(79) THEN LOCATE AKTZ,
80:GOTO 9390
[626] 9580 IF CSRLIN = RUEZ THEN LET ZUEN = 0
[0K7] 9590 IF ZUEN = 1 AND Z$ = CHR$(0) + CHR$(67) THEN
10000
[DG7] 9600 IF ZUEN = 1 AND Z$ = CHR$(0) + CHR$(81) THEN
10000
[EP3] 9610 IF ZUEN = 1 THEN 9390
[9GB] 9620 IF Z$ = CHR$(13) THEN LET ZA = 1:GOSUB 9900:
LET FUNK = 0:COLOR 7,0:RETURN
[DM9] 9630 IF Z$ = CHR$(0) + CHR$(65) AND EM = 0 THEN L
ET FUNK = 1:GOTO 9720
[NM9] 9640 IF Z$ = CHR$(0) + CHR$(66) AND EM = 0 THEN L
ET FUNK = 2:GOTO 9720
[XM9] 9650 IF Z$ = CHR$(0) + CHR$(67) AND EM = 0 THEN L
ET FUNK = 3:GOTO 9720
[968] 9660 IF Z$ = CHR$(0) + CHR$(68) THEN LET FUNK = 4
:GOTO 9720
[668] 9670 IF Z$ = CHR$(0) + CHR$(93) THEN LET FUNK = 5
:GOTO 9720
[HN3] 9680 IF EINF = 1 THEN 9830
[ZC5] 9690 IF LEFT$(Z$,1) = CHR$(0) THEN 9390
[NN8] 9700 IF ENDZ = 1 THEN PRINT Z$;:LOCATE AKTZ,80:GO
TO 9390
[VL3] 9710 PRINT Z$;:GOTO 9390
[0F3] 9720 LOCATE ,,0:RETURN
[8P0] 9730 '
[CP0] 9740 '
[H0A] 9750 REM Zeileninhalt rechts vom Cursor nach link
s schieben
[MP0] 9760 '
[KY8] 9770 LET MPOS = POS(0):LET ZA = POS(0):IF POS(0)
= 1 THEN 9390
[BSA] 9780 GOSUB 9930:PRINT CHR$(29);TEXT$;" ";:LOCATE
AKTZ,MPOS - 1:GOTO 9390
[0R0] 9790 '
[ZN0] 9800 '
[RG7] 9810 REM Zeichen in aktueller Zeile einfügen
[7P0] 9820 '
[C39] 9830 LET MPOS = POS(0):LET ZA = POS(0):IF POS(0)
= 80 THEN 9390
[1C7] 9840 GOSUB 9930:PRINT Z$;LEFT$(TEXT$,81 - POS(0))
;
[UD5] 9850 LOCATE AKTZ,MPOS + 1:GOTO 9390
[RP0] 9860 '
[VP0] 9870 '
[SY4] 9880 REM Zeileninhalt einlesen
[3R0] 9890 '
[UA2] 9900 LET ZZ = 81
[KK8] 9910 LET ZZ = ZZ - 1:IF ZZ = 0 THEN LET TEXT$ = "
":RETURN

```

```

[AR6] 9920 IF CHR$(SCREEN(AKTZ,ZZ)) = " " THEN 9910
[CK2] 9930 LET TEXT$ = ""
[ADB] 9940 FOR A = ZA TO ZZ:LET TEXT$ = TEXT$ + CHR$(SC
REEN(AKTZ,A)):NEXT:RETURN
[PP0] 9950 '
[UP0] 9960 '
[BK9] 9970 REM Textzeilen aus anderen Textstellen übern
ehmen
[2R0] 9980 '
[1R8] 9990 LET ZUEN = 1:LET RUEZ = CSRLIN:LET ZA = 1:GO
TO 9390
[3V6] 10000 GOSUB 9930:LOCATE RUEZ,1:PRINT TEXT$;
[FB4] 10010 LOCATE RUEZ,1:GOTO 9390
[0U0] 10020 '
[1U0] 10030 '
[3TA] 10040 REM Anfrage ob Zieldatei als aktuelle Datei
übernommen wird
[3U0] 10050 '
[U42] 10060 TIMER ON
[5R5] 10070 LET AS = INKEY$:IF AS = "" THEN 10070
[DK5] 10080 LET SPEICHER = FRE(""):TIMER OFF
[RK6] 10090 LOCATE 14,45:PRINT MN$(23):LOCATE 15,45
[NW9] 10100 PRINT MN$(24):LOCATE 16,45:PRINT MN$(25):LO
CATE 16,65:COLOR 8,7
[29A] 10110 PRINT MN$(26):COLOR 7,0:LOCATE 17,45:PRINT
MN$(27):LOCATE 18,45
[278] 10120 PRINT MN$(28):LOCATE 19,45:PRINT MN$(29):TI
MER ON
[ULA] 10130 LET AS = INKEY$:IF AS = CHR$(13) THEN LET D
ATEINAME$ = ZIEL$:RETURN
[S48] 10140 IF AS = "Q" OR AS = "q" OR AS = " " THEN RE
TURN ELSE 10130
[7U0] 10150 '
[8U0] 10160 '
[754] 10170 REM Anzeigen löschen
[AU0] 10180 '
[G27] 10190 FOR A = 6 TO 9:LOCATE A,45:PRINT SPACE$(35)
:NEXT
[G68] 10200 FOR A = 14 TO 19:LOCATE A,45:PRINT SPACE$(3
5):NEXT
[7A8] 10210 LOCATE 23,61:PRINT SPACE$(12);:GOSUB 11400:
RETURN
[Y0B] 10220 COLOR 7,0:FOR A = 16 TO 19:LOCATE A,45:PRIN
T SPACE$(35):NEXT:RETURN
[9U0] 10230 '
[AU0] 10240 '
[L3A] 10250 REM kleine Funktionsauswahl und Rückkehr zu
m Hauptmenü
[CU0] 10260 '
[3N9] 10270 IF MFN = 1 THEN LET XT = 24:LET YT = 70:GOS
UB 8420:GOTO 10380
[Z89] 10280 LET XT = 25:LET YT = 72:VIEW PRINT:LOCATE 2
5,1:COLOR 8,7
[E1A] 10290 PRINT " ";CHR$(3);" Menü = > ";:COLOR 16,7:
PRINT "Q";:COLOR 8,7
[5HB] 10300 PRINT " < ";:COLOR 7,0:PRINT CHR$(16);" ";:
COLOR 10,0:PRINT LWS;:COLOR 7,0
[BN6] 10310 PRINT " ";CHR$(17);CHR$(16);" ";:COLOR 10,0
[3X4] 10320 IF DATEINAME$ <> "" THEN 10340
[0U4] 10330 PRINT MN$(31);" ";:GOTO 10370
[5T9] 10340 PRINT DATEINAME$;" ";:COLOR 7,0:PRINT CHR$(
17);:COLOR 8,7
[WYA] 10350 PRINT " Lesen = > ";:COLOR 16,7:PRINT "L";:
COLOR 8,7:PRINT " < ";
[2AA] 10360 PRINT " Edit. = > ";:COLOR 16,7:PRINT "E";:
COLOR 8,7:PRINT " < ";
[5M6] 10370 COLOR 7,0:PRINT TAB(71) STRING$(10,219);
[BY3] 10380 LET SPEICHER = FRE("")
[0Y3] 10390 TIMER ON:GOSUB 11400
[1R5] 10400 LET AS = INKEY$:IF AS = "" THEN 10400
[1W3] 10410 GOSUB 8060:TIMER OFF
[HL4] 10420 IF AS = "Q" OR AS = "q" THEN 500
[985] 10430 IF AS = CHR$(0) + CHR$(15) THEN 500
[SS9] 10440 IF AS = CHR$(0) + CHR$(59) THEN LET DATEINA
ME$ = FS(1):GOTO 10270
[WR9] 10450 IF AS = CHR$(0) + CHR$(60) THEN LET DATEINA
ME$ = FS(2):GOTO 10270
[5S9] 10460 IF AS = CHR$(0) + CHR$(61) THEN LET DATEINA
ME$ = FS(3):GOTO 10270
[ES9] 10470 IF AS = CHR$(0) + CHR$(62) THEN LET DATEINA
ME$ = FS(4):GOTO 10270
[RS9] 10480 IF AS = CHR$(0) + CHR$(63) THEN LET DATEINA
ME$ = FS(5):GOTO 10270
[0T9] 10490 IF AS = CHR$(0) + CHR$(64) THEN LET DATEINA
ME$ = FS(6):GOTO 10270
[3T9] 10500 IF AS = CHR$(0) + CHR$(65) THEN LET DATEINA
ME$ = FS(7):GOTO 10270
[CT9] 10510 IF AS = CHR$(0) + CHR$(66) THEN LET DATEINA
ME$ = FS(8):GOTO 10270

```

Das Basic-Listing »bukamav.asc« ist nicht nur ein Texteditor


```

[NT9] 10520 IF A$ = CHR$(0) + CHR$(67) THEN LET DATEINA
ME$ = F$(9):GOTO 10270
[OU9] 10530 IF A$ = CHR$(0) + CHR$(68) THEN LET DATEINA
ME$ = F$(10):GOTO 10270
[HZ9] 10540 IF A$ = "L" OR A$ = "l" THEN LET LESEDATEI$
= DATEINAME$:GOTO 950
[W6A] 10550 IF A$ = "R" OR A$ = "r" THEN LET LESEDATEI$
= "RÖDEL.NED":GOTO 950
[NU9] 10560 IF A$ = "S" OR A$ = "s" THEN LET LESEDATEI$
= "GASED.NED":GOTO 950
[HY9] 10570 IF A$ = "T" OR A$ = "t" THEN LET LESEDATEI$
= "TIMOP.NED":GOTO 950
[G09] 10580 IF A$ = "Z" OR A$ = "z" THEN LET LESEDATEI$
= ZIEL$:GOTO 950
[Z09] 10590 IF A$ = "A" OR A$ = "a" THEN OPEN "I",#1,DA
TEINAME$:GOTO 1410
[709] 10600 IF A$ = "E" OR A$ = "e" THEN OPEN "I",#1,DA
TEINAME$:GOTO 2040
[MH9] 10610 IF A$ = CHR$(9) THEN CLS:SHELL "DIR/P":LET
MFN = 0:GOTO 10280
[SU4] 10620 IF A$ = "F" OR A$ = "f" THEN 5580
[K6A] 10630 IF A$ = "H" OR A$ = "h" THEN LET MFN = 0:LE
T MH = 1:GOTO 6540 ELSE 10400
[U00] 10640 '
[VU0] 10650 '
[ZT9] 10660 REM Funktionstastenbelegung anzeigen (Sonde
rzeichen)
[XU0] 10670 '
[B18] 10680 LOCATE 25,1:COLOR 7,0:PRINT CHR$(4);" ";:CO
LOR 8,7
[8N6] 10690 PRINT " F1 ";:COLOR 7,0:PRINT CHR$(16);
[U47] 10700 COLOR 10,0:PRINT " ";CHR$(F1);" ";:COLOR 8
,7
[5N6] 10710 PRINT " F2 ";:COLOR 7,0:PRINT CHR$(16);
[Z47] 10720 COLOR 10,0:PRINT " ";CHR$(F2);" ";:COLOR 8
,7
[8N6] 10730 PRINT " F3 ";:COLOR 7,0:PRINT CHR$(16);
[457] 10740 COLOR 10,0:PRINT " ";CHR$(F3);" ";:COLOR 8
,7
[BN6] 10750 PRINT " F4 ";:COLOR 7,0:PRINT CHR$(16);
[957] 10760 COLOR 10,0:PRINT " ";CHR$(F4);" ";:COLOR 8
,7
[EN6] 10770 PRINT " F5 ";:COLOR 7,0:PRINT CHR$(16);
[E57] 10780 COLOR 10,0:PRINT " ";CHR$(F5);" ";:COLOR 8
,7
[HN6] 10790 PRINT " F6 ";:COLOR 7,0:PRINT CHR$(16);
[D57] 10800 COLOR 10,0:PRINT " ";CHR$(F6);" ";:COLOR 8
,7
[8S6] 10810 PRINT " Ende der Eingabe = F10 ";:RETURN
[0V0] 10820 '
[1V0] 10830 '
[U69] 10840 REM Funktionstastenbelegung anzeigen (Datei
namen)
[3V0] 10850 '
[8B4] 10860 GOSUB 7380:LET MDAZG = 1
[FLB] 10870 LOCATE 14,45:COLOR 1,0:PRINT "Funktionstast
enbelegung/Dateinamen"
[NX9] 10880 COLOR 7,0:LOCATE 15,45:PRINT "1 ";:COLOR 8,
7:PRINT " ";F$(1);" ";
[WZ9] 10890 COLOR 7,0:LOCATE 15,61:PRINT " 6 ";:COLOR 8
,7:PRINT " ";F$(6);" ";
[RX9] 10900 COLOR 7,0:LOCATE 16,45:PRINT "2 ";:COLOR 8,
7:PRINT " ";F$(2);" ";
[Z0A] 10910 COLOR 7,0:LOCATE 16,61:PRINT " 7 ";:COLOR 8
,7:PRINT " ";F$(7);" ";
[ZX9] 10920 COLOR 7,0:LOCATE 17,45:PRINT "3 ";:COLOR 8,
7:PRINT " ";F$(3);" ";
[E0A] 10930 COLOR 7,0:LOCATE 17,61:PRINT " 8 ";:COLOR 8
,7:PRINT " ";F$(8);" ";
[7Y9] 10940 COLOR 7,0:LOCATE 18,45:PRINT "4 ";:COLOR 8,
7:PRINT " ";F$(4);" ";
[U0A] 10950 COLOR 7,0:LOCATE 18,61:PRINT " 9 ";:COLOR 8
,7:PRINT " ";F$(9);" ";
[FY9] 10960 COLOR 7,0:LOCATE 19,45:PRINT "5 ";:COLOR 8,
7:PRINT " ";F$(5);" ";
[54A] 10970 COLOR 7,0:LOCATE 19,61:PRINT "10 ";:COLOR
8,7:PRINT " ";F$(10);" ";
[8D3] 10980 COLOR 7,0:RETURN
[BV0] 10990 '
[1U0] 11000 '
[YW8] 11010 REM Funktionstastenbelegung anzeigen (Pfadn
amen)
[3U0] 11020 '
[8A4] 11030 GOSUB 7380:LET MDAZG = 1
[WDB] 11040 LOCATE 14,45:COLOR 1,0:PRINT "Funktionstast
enbelegung/Pfadnamen"
[LU9] 11050 COLOR 7,0:LOCATE 15,45:PRINT "1 ";:COLOR 8,
7:PRINT " ";F$(1);" ";
[62A] 11060 COLOR 7,0:LOCATE 15,61:PRINT " 6 ";:COLOR 8
,7:PRINT " ";F$(6);" "

```

```

[WU9] 11070 COLOR 7,0:LOCATE 16,45:PRINT "2 ";:COLOR 8,
7:PRINT " ";F$(12);" ";
[F2A] 11080 COLOR 7,0:LOCATE 16,61:PRINT " 7 ";:COLOR 8
,7:PRINT " ";F$(17);" ";
[5V9] 11090 COLOR 7,0:LOCATE 17,45:PRINT "3 ";:COLOR 8,
7:PRINT " ";F$(13);" ";
[K2A] 11100 COLOR 7,0:LOCATE 17,61:PRINT " 8 ";:COLOR 8
,7:PRINT " ";F$(18);" ";
[8V9] 11110 COLOR 7,0:LOCATE 18,45:PRINT "4 ";:COLOR 8,
7:PRINT " ";F$(14);" ";
[V2A] 11120 COLOR 7,0:LOCATE 18,61:PRINT " 9 ";:COLOR 8
,7:PRINT " ";F$(19);" ";
[HV9] 11130 COLOR 7,0:LOCATE 19,45:PRINT "5 ";:COLOR 8,
7:PRINT " ";F$(15);" ";
[93A] 11140 COLOR 7,0:LOCATE 19,61:PRINT "10 ";:COLOR
8,7:PRINT " ";F$(20);" ";
[8C3] 11150 COLOR 7,0:RETURN
[BU0] 11160 '
[CU0] 11170 '
[C0B] 11180 REM Abzählmaßstäbe anzeigen und Textfenster
einrichten
[EU0] 11190 '
[ER7] 11200 LET LINIE$ = STRING$(79,196) + CHR$(180):CO
LOR 10,0
[9M8] 11210 LOCATE 2,1:PRINT LINIE$:LOCATE 24,1:PRINT L
INIE$;
[HR3] 11220 FOR A = 5 TO 75 STEP 5
[G99] 11230 LOCATE 2,A:PRINT CHR$(194);:LOCATE 24,A:PRI
NT CHR$(193);
[UE1] 11240 NEXT
[818] 11250 COLOR 7,0:VIEW PRINT 3 TO 23:LOCATE 3,1:RET
URN
[FU0] 11260 '
[GU0] 11270 '
[0U5] 11280 REM Dateiinhalte transferieren
[KU0] 11290 '
[4N3] 11300 LET Z = 0:LET TZ = 0
[RN8] 11310 IF NOT EOF(1) THEN LINE INPUT #1,TEXT$(Z) E
LSE 11340
[K88] 11320 IF LEN(TEXT$) > 80 THEN LET TEXT$ = LEFT$(T
EXT$,80)
[Z95] 11330 LET Z = Z + 1:IF Z < 50 THEN 11310
[M1A] 11340 IF TZ < Z THEN PRINT #2,TEXT$(TZ):LET TZ =
TZ + 1:GOTO 11340
[556] 11350 IF EOF(1) = -1 THEN RETURN ELSE 11300
[L00] 11360 '
[MU0] 11370 '
[E66] 11380 REM Funktionstasten inaktivieren
[P00] 11390 '
[B56] 11400 FOR A = 1 TO 10:KEY A,"":NEXT:RETURN
[KU0] 11410 '
[L00] 11420 '
[TK7] 11430 REM Funktionstastenbelegung (Dateinamen)
[NU0] 11440 '
[KG6] 11450 FOR A = 1 TO 10:KEY A,F$(A):NEXT:RETURN
[RU0] 11460 '
[SU0] 11470 '
[AB7] 11480 REM Funktionstastenbelegung (Pfadnamen)
[U00] 11490 '
[YB7] 11500 FOR A = 11 TO 20:KEY A - 10,F$(A):NEXT:RETU
RN
[P00] 11510 '
[RU0] 11520 '
[TS7] 11530 REM Funktionstastenbelegung (ASCII - Codes)
[TU0] 11540 '
[HY6] 11550 KEY 1,CHR$(F1):KEY 2,CHR$(F2):KEY 3,CHR$(F3
)
[A88] 11560 KEY 4,CHR$(F4):KEY 5,CHR$(F5):KEY 6,CHR$(F6
):RETURN
[WU0] 11570 '
[XU0] 11580 '
[SF9] 11590 REM Kommandoprozessor suchen und Bildschirm
löschen
[TU0] 11600 '
[DM5] 11610 SHELL "CLS":LOCATE 1,1,0:RETURN
[VU0] 11620 '
[WU0] 11630 '
[YN9] 11640 REM Einschaltzustandsanzeige für akustische
s Signal
[YU0] 11650 '
[LT6] 11660 IF TON = 1 THEN COLOR 8,7 ELSE COLOR 7,0
[U66] 11670 LOCATE 23,76:PRINT " ";CHR$(14);" ";:RETURN
[1V0] 11680 '
[ZV0] 11690 '
[UZ5] 11700 REM akustisches Signal ausgeben
[YU0] 11710 '
[184] 11720 IF TON = 0 THEN RETURN
[UX7] 11730 SOUND 349.23,6:SOUND 440,6:SOUND 523.25,6:R
ETURN
[1V0] 11740 '
[ZV0] 11750 '

```



```
[PV5] 11760 REM Anzeige der Fehlermeldungen
[4V0] 11770 '
[7C5] 11780 TIMER OFF:IF TON = 1 THEN BEEP
[R68] 11790 IF SCREEN(25,26) = 3 THEN COLOR 18,7 ELSE C
OLOR 16,7
[VY9] 11800 FOR A = 1 TO 4:READ FMZ$(A):LOCATE A + 5,45
:PRINT FMZ$(A):NEXT
[0D3] 11810 COLOR 7,0:RETURN
[3V0] 11820 '
[4V0] 11830 '
[T44] 11840 REM Fehlerbehandlung
[6V0] 11850 '
[RW7] 11860 IF ERL = 200 THEN LET LW$ = CHR$(4) + ">":R
ESUME 240
[R98] 11870 IF ERL = 290 OR ERL = 300 OR ERL = 310 THEN
RESUME 370
[N49] 11880 IF ERL = 7000 OR ERL = 7010 THEN RESTORE 13
070:RESUME 12110
[DM9] 11890 IF ERL = 11610 OR ERL = 4290 THEN RESTORE 1
3070:RESUME 12110
[S59] 11900 IF ERL = 4720 OR ERL = 4800 THEN RESTORE 13
070:RESUME 12110
[87B] 11910 IF ERL = 3890 OR ERL = 3900 OR ERL = 3910 T
HEN RESTORE 13070:RESUME 12110
[27B] 11920 IF ERL = 3920 OR ERL = 3930 OR ERL = 3940 T
HEN RESTORE 13070:RESUME 12110
[S7B] 11930 IF ERL = 3950 OR ERL = 3960 OR ERL = 3970 T
HEN RESTORE 13070:RESUME 12110
[BE7] 11940 IF ERL = 10610 THEN RESTORE 13070:RESUME 12
110
[1N4] 11950 IF ERR = 52 THEN RESUME 500
[537] 11960 IF ERR = 64 THEN RESTORE 12550:RESUME 12110
[L37] 11970 IF ERR = 53 THEN RESTORE 12590:RESUME 12110
[C37] 11980 IF ERR = 61 THEN RESTORE 12670:RESUME 12110
[937] 11990 IF ERR = 71 THEN RESTORE 12750:RESUME 12110
[H27] 12000 IF ERR = 72 THEN RESTORE 12790:RESUME 12110
[E17] 12010 IF ERR = 70 THEN RESTORE 12710:RESUME 12110
[5DA] 12020 IF ERR = 27 OR ERR = 25 OR ERR = 24 THEN RE
STORE 12870:RESUME 12110
[327] 12030 IF ERR = 58 THEN RESTORE 12630:RESUME 12110
[KR8] 12040 IF ERR = 75 OR ERR = 76 THEN RESTORE 12910:
RESUME 12110
[N27] 12050 IF ERR = 67 THEN RESTORE 12950:RESUME 12110
[137] 12060 IF ERR = 74 THEN RESTORE 12990:RESUME 12110
[TK4] 12070 RESTORE 13030:RESUME 12110
[6U7] 12080 IF SCREEN(6,45) = 32 THEN LET LW$ = MLW$:RE
TURN
[0Z5] 12090 RESTORE 12830:GOSUB 11780:TIMER ON
[TM9] 12100 LET A$ = INKEY$:IF A$ = "" THEN 12100 ELSE
GOSUB 10190:GOTO 3860
[EZ3] 12110 GOSUB 11780:TIMER ON
[ZP5] 12120 LET A$ = INKEY$:IF A$ = "" THEN 12120
[7C3] 12130 TIMER OFF:RESET
[NFA] 12140 IF SCREEN(25,26) <> 3 THEN CLS:GOTO 10280 E
LSE GOSUB 10190:GOTO 780
[DU0] 12150 '
[EU0] 12160 '
[8B6] 12170 REM Programmdaten (Menüanzeigen)
[GU0] 12180 '
[PW5] 12190 DATA " Textdatei lesen "
[NL6] 12200 DATA " Neue Textdatei schreiben "
[FU6] 12210 DATA " Text an Textdatei anfügen "
[K37] 12220 DATA " Dateien miteinander verbinden "
[7N6] 12230 DATA " Textdatei duplizieren "
[EA5] 12240 DATA " Editieren "
[4X5] 12250 DATA " Datei umbenennen "
[LU5] 12260 DATA " Dateien kopieren "
[XS5] 12270 DATA " Dateien löschen "
[A76] 12280 DATA " Inhaltsverzeichnis "
[287] 12290 DATA " Laufwerk/Verzeichnis wechseln "
[127] 12300 DATA " Unterverzeichnis erstellen "
[CK6] 12310 DATA " Disketten formatieren "
[N66] 12320 DATA " Textdatei drucken "
[WZ6] 12330 DATA " Funktionstasten neubelegen "
[HV7] 12340 DATA " Textdateigröße feststellen "
[796] 12350 DATA " Dokumentation/Hilfe "
[TP7] 12360 DATA " Ende/Rückkehr zum Betriebssystem "
[123] 12370 DATA "> B U K A M A V <"
[CP4] 12380 DATA "Geben Sie den Namen"
[BA5] 12390 DATA "Die Konvertierung der"
[ZD4] 12400 DATA "> Bitte warten <"
[3L6] 12410 DATA "Wenn Sie die Zieldatei als"
[FB7] 12420 DATA "aktuelle Datei übernehmen wollen,"
[U56] 12430 DATA "betätigen Sie Bitte"
[2B3] 12440 DATA "> RETURN <"
[3A4] 12450 DATA " "
[VW6] 12460 DATA "oder Sie kehren mit > Q < oder"
[3Y7] 12470 DATA "Leertaste zum Hauptprogramm zurück."
[LH6] 12480 DATA "Bitte wählen Sie eine Option."
[RW6] 12490 DATA "Keine Textdatei in Bearbeitung."
```

```
[XD5] 12500 DATA "ein Duplikat unter dem"
[TU0] 12510 '
[UU0] 12520 '
[R6] 12530 REM Programmdaten (Fehlermeldungen)
[WU0] 12540 '
[Z6] 12550 DATA "Der angegebene Dateiname "
[L56] 12560 DATA "entspricht nicht den "
[576] 12570 DATA "MS-DOS - Konventionen!"
[3L6] 12580 DATA "(unzulässige Zeichen)"
[817] 12590 DATA "Die gewählte Datei existiert "
[H37] 12600 DATA "nicht im aktuellen Verzeichnis!"
[7A4] 12610 DATA " "
[HW7] 12620 DATA "(Inhaltsverzeichnis überprüfen)"
[457] 12630 DATA "Es existiert bereits eine Datei "
[VF6] 12640 DATA "mit dem gleichen Namen im "
[6F6] 12650 DATA "aktuellen Verzeichnis!"
[CA4] 12660 DATA " "
[BB7] 12670 DATA "Ihre Diskette oder Festplatte ist "
[G17] 12680 DATA "voll. Wechseln Sie die Diskette. "
[K57] 12690 DATA "Bei Festplatte: räumen Sie das "
[WC5] 12700 DATA "Ding mal auf."
[847] 12710 DATA "Sie versuchen auf eine schreibge-"
[EP7] 12720 DATA "schützte Datei oder Diskette zu "
[675] 12730 DATA "schreiben."
[T66] 12740 DATA "> Zugriff abgelehnt <"
[NL7] 12750 DATA "Die Laufwerksverriegelung an einem"
[T17] 12760 DATA "Diskettenlaufwerk ist nicht "
[M57] 12770 DATA "geschlossen oder es ist keine "
[W66] 12780 DATA "Diskette eingelegt."
[CY6] 12790 DATA "Störung im Laufwerk oder "
[0Z7] 12800 DATA "die Diskette wurde entfernt,"
[H67] 12810 DATA "ausgewechselt oder beschädigt!"
[GA4] 12820 DATA " "
[Z27] 12830 DATA "Sie haben ein falsches bzw. nicht "
[MH7] 12840 DATA "existierendes Laufwerk gewählt!"
[7F7] 12850 DATA "Das aktuelle Laufwerk wurde nicht "
[DF5] 12860 DATA "gewechselt."
[4F6] 12870 DATA "Der Drucker ist nicht "
[2X6] 12880 DATA "eingeschaltet oder das Papier "
[PF6] 12890 DATA "ist zur Neige gegangen!"
[KA4] 12900 DATA " "
[TZ6] 12910 DATA "Sie haben einen Pfad angegeben, "
[BE7] 12920 DATA "welcher nicht existiert oder auf "
[8C7] 12930 DATA "den nicht zugegriffen werden kann."
[PA4] 12940 DATA " "
[EV7] 12950 DATA "Sie versuchen eine zu große Anzahl"
[L17] 12960 DATA "Dateien auf Ihren Datenträger zu "
[GS6] 12970 DATA "erstellen oder Sie geben eine "
[PH7] 12980 DATA "ungültige Dateispezifikation an."
[TT6] 12990 DATA "Sie haben unterschiedliche "
[837] 13000 DATA "Laufwerke beim Dateiumbenennen "
[F55] 13010 DATA "angegeben."
[L94] 13020 DATA " "
[W27] 13030 DATA "Innerhalb der BUKAMAV - Programm-"
[K57] 13040 DATA "datei ist ein Fehler aufgetreten,"
[U67] 13050 DATA "den Sie nicht zu vertreten haben."
[877] 13060 DATA "(Bitte Programmautor verständigen)"
[WM7] 13070 DATA "Der Kommandoprozessor wurde nicht "
[6M6] 13080 DATA "gefunden oder es liegt eine "
[BH7] 13090 DATA "Störung im Betriebssystem vor!"
[N94] 13100 DATA " "
[CU0] 13110 '
[DU0] 13120 '
[EH6] 13130 REM Programmdaten (Dokumentation)
[FU0] 13140 '
[TD8] 13150 DATA "Mit Hilfe dieser Einführung erhalten "
S"
[Y48] 13160 DATA "ie alle notwendigen Informationen zur "
[HT8] 13170 DATA "Benutzung des »BUKAMAV« - Text- und D "
a"
[HA7] 13180 DATA "teilverwaltungsprogramms."
[YL4] 13190 DATA " "
[TL4] 13200 DATA " "
[M88] 13210 DATA "Zum Auswählen der 18 Hauptfunktionen "
i"
[L78] 13220 DATA "m Zentralmenü können Sie die Cursor-"
[WL7] 13230 DATA "tasten, die Leer-, Tabulator- und die "
[B18] 13240 DATA "Backspacetaste benutzen. Weiterhin be "
-"
[318] 13250 DATA "steht die Möglichkeit mit Hilfe der P "
g"
[BN7] 13260 DATA "Up - und der Pg Dn - Taste schneller "
"
```

Das Basic-Listing »bukamav.asc« ist nicht nur ein Texteditor

[SB8] 13270 DATA "innerhalb der Hauptfunktionen herumzu
s"

[L78] 13280 DATA "pringen. Die endgültige Wahl erfolgt

[Z98] 13290 DATA "von einigen Ausnahmen abgesehen mit d
e"

[K06] 13300 DATA "r RETURN - Taste.

[VE8] 13310 DATA "Durch Betätigen von Funktionstaste 1
k"

[FA8] 13320 DATA "önnen Sie sich unter der Überschrifts
_"

[D08] 13330 DATA "zeile das aktuelle Verzeichnis anzeig
e"

[VZ7] 13340 DATA "n lassen. Eine erneute Betätigung die
_"

[VT7] 13350 DATA "ser Taste schaltet die Anzeige wieder

[HX7] 13360 DATA "ab. Den Namen der aktuellen Textdatei

[C18] 13370 DATA "sowie der zum Drucken bereitgestellten

[EY7] 13380 DATA "Datei und der aktuelle Zielpfad werde
n"

[368] 13390 DATA "mit Funktionstaste 2 angezeigt. Die F
u"

[9W7] 13400 DATA "nktionstasten 3 und 4 dienen zum An-

[ZY7] 13410 DATA "zeigen der voreingestellten Datei- und
d"

[F48] 13420 DATA "Pfadnamen. Die Tonerzeugung schalten

[B38] 13430 DATA "Sie mit Funktionstaste 5 ein und mit
F"

[CX7] 13440 DATA "unktionstaste 6 wieder aus. Die Funk-

[1X7] 13450 DATA "tionstaste 10 löscht alle Anzeigen de
r"

[YW6] 13460 DATA "Funktionstasten 2, 3 und 4.

[M48] 13470 DATA "Das aktuelle Laufwerk wird ständig im

[ZV7] 13480 DATA "rechten unterem Feld neben dem Eingab
e"

[HT7] 13490 DATA "befeld angezeigt. Das Anzeigefeld gan
z"

[3A8] 13500 DATA "rechts unten gibt Auskunft darüber o
b"

[H68] 13510 DATA "die Tonerzeugung eingeschaltet ist od
e"

[8B5] 13520 DATA "r nicht.

[D18] 13530 DATA "Im Folgenden werden alle » BUKAMAV «
_"

[8E7] 13540 DATA "Hauptfunktionen dargestellt:

[AM4] 13550 DATA "

[BM4] 13560 DATA "

[676] 13570 DATA " --- Textdatei lesen ---

[DM4] 13580 DATA "

[7Z7] 13590 DATA "Nachdem Sie mit der RETURN - Taste di
e"

[NV7] 13600 DATA "se Funktion gewählt haben, geben Sie

[488] 13610 DATA "Bitte den gewünschten Dateinamen ein
u"

[5T7] 13620 DATA "nd bestätigen mit der RETURN - Taste.

[S48] 13630 DATA "Wenn diese Datei im aktuellen Verzeic
h"

[L28] 13640 DATA "nis existiert, wird sie seitenweise

[6Y7] 13650 DATA "auf dem Bildschirm ausgegeben. Alles
w"

[5B8] 13660 DATA "eitere erfolgt menügesteuert und er-

[548] 13670 DATA "fordert keine weiteren Erklärungen.

[HM4] 13680 DATA "

[M38] 13690 DATA "Die aktuelle Datei kann jeweils durch

[MN7] 13700 DATA "Betätigung der Taste » A < aufgerufen

[498] 13710 DATA "werden (ohne Bestätigung mit RETURN).

[R18] 13720 DATA "Das Gleiche gilt bei Betätigung der

[EA7] 13730 DATA "Tasten » R <, » S < und » T < zum Les
e"

[E08] 13740 DATA "n der temporären Dateien » RÖDEL.NED
<"

[SF7] 13750 DATA "» GASED.NED < und » TIMOP.NED < und »

[L18] 13760 DATA "Z < für eine eventuelle Zieldatei.

[MM4] 13770 DATA "

[NM4] 13780 DATA "

[K47] 13790 DATA " --- Neue Textdatei schreiben ---

[HM4] 13800 DATA "

[6A8] 13810 DATA "Die Eingabe des Dateinamens erfolgt w
i"

[UK7] 13820 DATA "e bei » Textdatei lesen <. Wenn eine

[CW7] 13830 DATA "Datei mit gleichem Namen im aktuellen

[GX7] 13840 DATA "Verzeichnis existiert und es liegt

[Z68] 13850 DATA "kein Schreibschutzattribut vor, dann
w"

[T68] 13860 DATA "ird sie ohne Warnung überschrieben.

[7K8] 13870 DATA "Anschließend wird die Bildschirmanzei
g"

[098] 13880 DATA "e gewechselt und Sie können eine neue

[888] 13890 DATA "Textdatei schreiben. Die Länge der Te
x"

[SX7] 13900 DATA "tzeilen ist auf 80 Zeichen begrenzt.

[1R8] 13910 DATA "Jede Textzeile muß mit RETURN bestäti
g"

[938] 13920 DATA "t werden. Der Einfügemodus wird mit

[S78] 13930 DATA "der Ins - Taste ein- und ausgeschalte
t"

[4T7] 13940 DATA ". Mit der Backspacetaste wird der Zei
_"

[9A8] 13950 DATA "leninhalt rechts vom Cursor nach link
s"

[9Y7] 13960 DATA "geschoben. Wenn Sie den Inhalt einer

[B68] 13970 DATA "anderen auf dem Bildschirm angezeigte
n"

[7U7] 13980 DATA "Zeile in die aktuelle Zeile überneh
e"

[848] 13990 DATA "men wollen, müssen Sie zuerst die Pg
U"

[GW7] 14000 DATA "p - Taste betätigen und dann mit der

[VZ7] 14010 DATA "Cursortaste in die fragliche Zeile wa
n"

[RL7] 14020 DATA "dern und mit der Pg Dn - Taste ausle
_"

[5C8] 14030 DATA "sen. Anschließend kann die aktuelle Z
e"

[187] 14040 DATA "ile weiter bearbeitet werden.

[F98] 14050 DATA "Die Texteingabe wird mit der Funktion
s"

[EX5] 14060 DATA "taste 10 beendet.

[6A8] 14070 DATA "Alles Weitere erfolgt menügesteuert.

[WL4] 14080 DATA "

[XL4] 14090 DATA "

[SL4] 14100 DATA "

[W97] 14110 DATA " --- Text an Textdatei anfügen ---

[UL4] 14120 DATA "

[RW7] 14130 DATA "Der Aufruf und die Texteingabe erfolg
t"

[4T7] 14140 DATA "wie bei » Neue Textdatei schreiben <

[UC7] 14150 DATA "(kann auch durch die Eingabe von » A
<"

[R88] 14160 DATA "in den kleinen Menüs gewählt werden)


```
[2R7] 14170 DATA "Es wird am Anfang die letzte Zeile de
[5W7] 14180 DATA " aufgerufenen Textdatei angezeigt.
[1M4] 14190 DATA "
[WL4] 14200 DATA "
[OK7] 14210 DATA " --- Dateien miteinander verbinden --
[YL4] 14220 DATA "
[DV7] 14230 DATA "Der Aufruf erfolgt wie gewohnt, alles
[6A8] 14240 DATA "Weitere erfolgt menügesteuert. Es wir
[FV7] 14250 DATA "der Inhalt einer anderen Textdatei an
[GK7] 14260 DATA "die aufgerufene Datei angefügt.
[628] 14270 DATA "Die zweite aufgerufene Datei bleibt u
[NN5] 14280 DATA "verändert.
[MV6] 14290 DATA " --- Textdatei duplizieren ---
[OM4] 14300 DATA "
[FV7] 14310 DATA "Der Aufruf erfolgt wie gewohnt, alles
[8A8] 14320 DATA "Weitere erfolgt menügesteuert. Es wir
[Z98] 14330 DATA "ein identisches Duplikat der aufgeruf
[D37] 14340 DATA "nen Textdatei angefertigt.
[5M4] 14350 DATA "
[6M4] 14360 DATA "
[2R5] 14370 DATA " --- Editieren ---
[8M4] 14380 DATA "
[5X7] 14390 DATA "Der Aufruf und die Texteingabe erfolg
[H68] 14400 DATA " wie gewohnt. Das Einfügen und Lösche
[EK8] 14410 DATA "von Textzeilen erfolgt menügesteuert.
[E48] 14420 DATA "Wenn Sie zur Beendigung der Textbear-
[VU7] 14430 DATA "beitung die Shifttaste und F10 gleich
[MP7] 14440 DATA "eitig drücken, wird die Datei ab der
[3W7] 14450 DATA "aktuellen Zeile abgetrennt und in der
[228] 14460 DATA "Datei > RÖDEL.NED < gespeichert. Wird
[T58] 14470 DATA "infolge eines Hardwarefehlers die Tex
[818] 14480 DATA "bearbeitung abgebrochen, kann ein Tei
[DX7] 14490 DATA "der Daten mit Hilfe der Datei > TIMOP
[BF6] 14500 DATA "NED < gerettet werden.
[W18] 14510 DATA "Die Beendigung der Konvertierung wird
[VX7] 14520 DATA "gegebenenfalls durch ein Tonsignal an
[UF5] 14530 DATA "gezeigt.
[CM4] 14540 DATA "
[DM4] 14550 DATA "
[EM4] 14560 DATA "
[9W6] 14570 DATA " --- Inhaltsverzeichnis ---
[GM4] 14580 DATA "
[9M8] 14590 DATA "Außer dem gewohnten Aufruf besteht no
[728] 14600 DATA "h die Möglichkeit durch Betätigung de
[D68] 14610 DATA "entsprechenden Tasten die Laufwerke d
```

```
[8V7] 14620 DATA "rekt abzufragen. Wenn Sie die Tasten
[6H7] 14630 DATA "> P < oder > S < betätigen, wird das
[XV7] 14640 DATA "ktuelle Inhaltsverzeichnis in der
[WW7] 14650 DATA "Datei > GASED.NED < protokolliert und
[YA7] 14660 DATA "kann weiter bearbeitet werden.
[3T7] 14670 DATA " --- Laufwerk/Verzeichnis wechseln --
[MM4] 14680 DATA "
[9C8] 14690 DATA "Die Laufwerke werden durch Betätigung
[3A8] 14700 DATA "der entsprechenden Taste angewählt.
[C78] 14710 DATA "Die Unterverzeichnisse werden durch E
[1G7] 14720 DATA "ngabe des Pfadnamens gewechselt.
[LM4] 14730 DATA "
[MM4] 14740 DATA "
[5N7] 14750 DATA " --- Funktionstasten neubelegen ---
[PM4] 14760 DATA "
[LE8] 14770 DATA "Zur Belegung der Funktionstasten mit
[1V7] 14780 DATA "onderzeichen betätigen Sie RETURN.
[4Z7] 14790 DATA "Mit Ctrl - Funktionstaste belegen Sie
[E88] 14800 DATA "die entsprechende Funktionstaste mit
[RW7] 14810 DATA "einem Dateinamen und mit Alt - Funkti
[H28] 14820 DATA "nstaste mit einem Pfadnamen. Die Tas-
[S38] 14830 DATA "tenbelegung kann auch mit Hilfe der K
[NY7] 14840 DATA "nfigurationsdatei > BUKAMAV.NED < in
[D78] 14850 DATA "der Reihenfolge Sonderzeichen, Datein
[F47] 14860 DATA "men und Pfadnamen erfolgen.
[VM4] 14870 DATA "
[WM4] 14880 DATA "
[FG8] 14890 DATA "Die übrigen Funktionen sind alle menü
[388] 14900 DATA "esteuert und erfordern keine weiteren
[FB8] 14910 DATA "Erläuterungen. Bedienungsfehler und F
[ZD8] 14920 DATA "lscheingaben werden mit entsprechende
[EC8] 14930 DATA "Fehlermeldungen quittiert und führen
[T48] 14940 DATA "ormalerweise nicht zu einem Programm-
[S37] 14950 DATA "abbruch oder Datenverlust.
[YM4] 14960 DATA "
[ZM4] 14970 DATA "
[ON4] 14980 DATA "
[M08] 14990 DATA "Der Name > BUKAMAV < stammt aus der W
[ZY7] 15000 DATA "lthilfssprache Volapük und bedeutet
[AA8] 15010 DATA "soviel wie Buchführung und im übertra
[F97] 15020 DATA "enem Sinne Textverarbeitung.
[HD7] 15030 DATA " --- Copyright (c) by Wilhelm Andreas
[RL6] 15040 DATA " Im Östern 6, 4690 Herne 2 ---
```

Das Basic-Listing »bukamav.asc« ist nicht nur ein Texteditor

Raumschiff Stardos

– bitte melden!

Die Erde ist überbevölkert. Es gibt keine Nahrung mehr, allein neue Kolonien auf Himmelskörpern inner- und außerhalb unseres Sonnensystems können noch Rettung bringen. Mit dem Computer »Argemonia« lernen Sie aber schon auf der Erde, die gefährlichsten Situationen bei der Entdeckung neuer Welten kennen. Spielen Sie mit bei dieser spannenden Weltraumsimulation!

Wir schreiben das Jahr 2069. Die Zahl der Erdbewohner hat sich seit Beginn des Jahrhunderts mehr als verfünffacht. Noch nie mußte der blaue Planet so vielen Menschen Platz bieten und sie ernähren. Nahrungsmittel und Rohstoffe gibt es kaum mehr. Kriege um die letzten Ressourcen sind an der Tagesordnung. Nur noch ein Gedanke erfüllt die Menschen, das pure Überleben. Als einzige Chance bleibt die Erforschung fremder Welten, um diese zu besiedeln.

Insgesamt 18 Planeten sind von den Wissenschaftlern der Vereinten Nationen für eine intensivere Untersuchung ausgewählt worden. Dort besteht Hoffnung, eine lebensgerechte Umwelt zu finden. Doch eine Reise zu den Sternen ist auch im 21. Jahrhundert noch gefährlich und fordert viele Opfer. So wurde der Trainingscomputer »Argemonia« entwickelt, auf dem die Zukunft auf den 18 Planeten simuliert wird. Die Auswanderer gewöhnen sich so schon zu Hause an die feindliche Umwelt.

Es gibt wieder Probleme auf der guten alten Erde: – Die Suche nach einem neuen Planeten beginnt

Das Weltraumspiel »Argemonia« umfaßt in insgesamt 54 Szenen drei verschiedene Spiele. Die Handlung der Ebene A (Bild 1) ähnelt dem Spiel Q-Bert. Ebene B (Bild 2) ist eine Kombination zwischen Asteroids und Worms. Als letztes agieren Sie in der Ebene C (Bild 3), wo Sie feindliche Raumschiffe an Landung auf dem Planeten hindern müssen.

Nach dem Programmstart sehen Sie das Titelbild (Bild 4) auf Ihrem Bildschirm und steuern mit drei Tasten das weitere Geschehen. Mit [E] verlassen Sie das Programm und die Systemebene von MS-DOS wird wieder aufgerufen. [R] drückt die Spielregeln (Bild 5) aus und [S] startet das Spiel.

Ihre Aufgabe in der ersten Ebene ist es, alle Blockoberflächen einzufärben. Dazu müssen Sie Ihre Spielfigur auf einen Block steuern, der sich dann verfärbt. Allerdings dürfen Sie auf keinen Fall mit den Gegnern in Berührung kommen – denn sonst verlieren Sie eines Ihrer Leben. Die Spielfigur steuern Sie mit den Tasten [1] (Sprung nach unten links), [3] (Sprung nach unten rechts), [4] (Schritt nach links), [6] (Schritt nach rechts), [7] (Sprung nach oben links) und [9] (Sprung nach oben rechts) auf dem Tastenblock. Die NumLock-Taste darf dabei allerdings nicht gedrückt sein.

Mit [S] wird ein Supersprung ausgeführt und die Spielfigur auf einen beliebigen anderen Block »gebeamt«. Da das Ziel zufällig ist, und der Supersprung 50 Punkte kostet, empfiehlt sich dieser Supersprung nur als Rettung aus höchster Not. Sind alle Steine grün eingefärbt, geht das Spiel auf dem gleichen Planeten in der Ebene B weiter.

Um die Struktur des Himmelskörpers festzustellen, müssen Sie verschiedene Gesteinsproben sammeln, wiederum ohne von den wild umherfliegenden Asteroiden getroffen zu werden. Ihr Fahrzeug steuern Sie mit den Tasten [2] (abwärts), [4] (links), [6] (rechts) und [8] (aufwärts). Weichen

Sie allem aus, was sich bewegt – sonst ist Ihr Leben keinen Pfifferling mehr wert.

Zum Einsammeln der Gesteinsproben steuern Sie die Kästen in der Reihenfolge ihrer Nummern an und räumen sie aus. Dazu rammen Sie die Kästen einfach an einer Seite. Treffen Sie allerdings auf eine Ecke oder rammen Sie einen Block, der nicht an der Reihe ist, so ist wieder einmal eines Ihrer Leben verloren. Sind alle Blöcke leer, so haben Sie auch diese Aufgabe erfüllt und es geht weiter in die Ebene C.

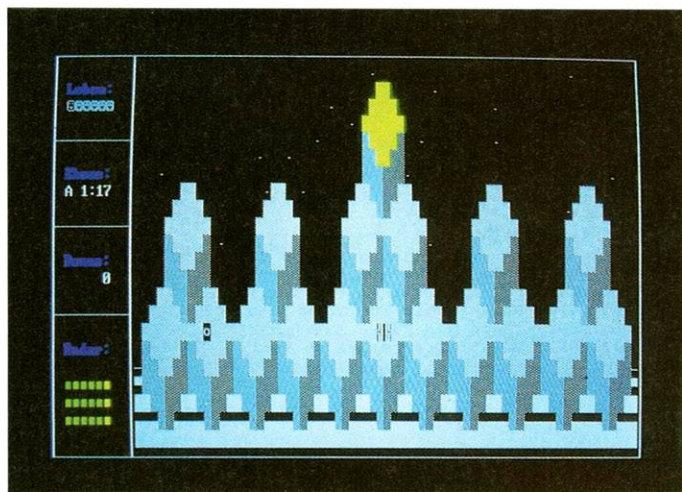
Jetzt sind Sie Ihres Lebens sicher. In der Bonusebene C können Sie nämlich kein Leben verlieren – aber dafür ein weiteres hinzugewinnen. Ihre Aufgabe ist es, Ihr Schiff in der unteren Bildschirmzeile mit den Tasten [4] (links) und [6] (rechts) unter die landenden Raumschiffe zu steuern und diese mit [S] abzuschießen. Sind allerdings zehn der Schiffe gelandet, so können Sie keine Punkte mehr sammeln. Sie fliegen – um ein weiteres Leben reicher – zum nächsten Planeten, wo das Abenteuer in Ebene A mit einer anderen Schwierigkeitsstufe von vorn beginnt.

Ein Highscore verrät Ihnen, wie erfolgreich Sie die Abenteuer überstanden haben

Sind alle Leben verbraucht, so wird der Punktestand bewertet. Falls Sie zu den zehn besten Spielern gehören, wird Ihnen die Ehre zuteil, in die Highscore-Tabelle aufgenommen zu werden. Wie die Punkte gezählt werden, sehen Sie in der Tabelle. In allen anderen Fällen starten Sie mit einer beliebigen Taste eine neue Runde.

Argemonia ist ein Basic-Programm. Es läuft unter GW-Basic und den beiden Compilern Turbo Basic und Quick Basic. Eine Grafikkarte ist nicht notwendig, da der Computer

Bild 1. In der Ebene A ist schnelles Reagieren gefragt



allein im Textmodus betrieben wird. Beim Programmstart muß die Programm diskette, die das compilierte Programm oder den GW-Basic-Interpreter mit dem Listing enthält, im Laufwerk A liegen. Ferner muß sich eine Datei für den Highscore auf dieser Diskette befinden.

Diese Datei geben Sie direkt auf der Systemebene vor. Tippen Sie

copy con: arge.hi

und die Zeilen aus Listing 1. Sie schließen jede Zeile mit [Enter] ab. Zum Schluß drücken Sie [F6] und [Enter]. Damit steht die Highscoretafel auf der Diskette.

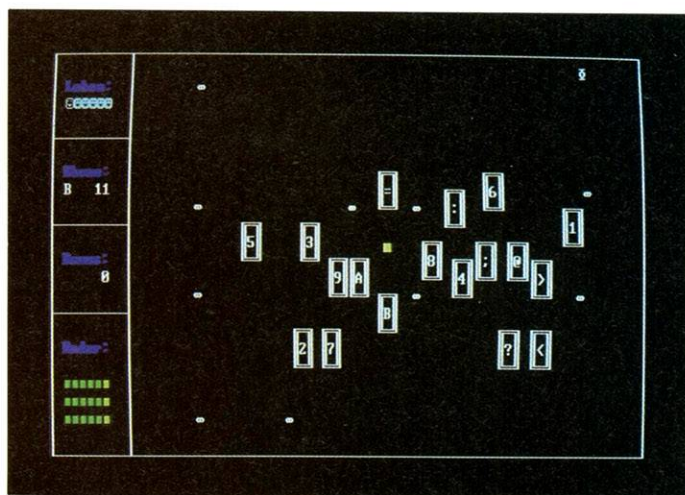
Als nächstes geben Sie das Programm aus Listing 2 ein. Es beginnt mit der Autorenmeldung. Danach wird in den Zeilen 180 bis 300 der Startzustand des Computers festgelegt. Dieser ist zwar beim Aufruf des GW-Basic-Interpreters automatisch aktiv, nicht aber, wenn Sie ein anderes Programm benutzt haben, das Daten der Bildschirmfarbe, Cursorposition und ähnliches verändert hat. In der Zeile 300 erfolgt automatisch eine Anpassung des Programms an eine monochrome Hercules- oder CGA-Grafikkarte. Der Unterschied dieser beiden Karten liegt in der Startadresse des Bildspeichers (bei CGA B800hex und bei Hercules B000hex).

Auch in der Zukunft braucht man Technik – Argemonia ist vollkommen in Basic programmiert

In dieser Initialisierungsroutine werden ferner die Startadresse für die Fehleroutine und die Startziffer für den Zufallsgenerator (in Abhängigkeit von der Zeit, die der Computer eingeschaltet ist) festgelegt. Der Bezug auf die Zeit erzeugt eine nahezu völlig zufällige Zahlenverteilung, da dieser Wert sich von Sekunde zu Sekunde ändert und damit nur sehr selten doppelt auftaucht.

Der dritte Programmteil legt sämtliche Variablen für das Spiel fest. Da die Bildschirmmeldungen als Zeichenketten ausgegeben werden, stehen auch diese Informationen hier. Ab Zeile 1070 wird der Rahmen für das Titelbild gezeichnet. Da dies mit Print-Anweisungen sehr viel Zeit erfordern würde, wird mit dem Befehl »poke« direkt in den Bildspeicher geschrieben. Insgesamt 4000 Speicherplätze stehen dazu zur Verfügung. 80 Zeichen in 25 Zeilen benötigen 2000 Speicherzellen, in den anderen 2000 Zellen stehen die zugehörigen Farbattribute. So lassen sich sämtliche ASCII-Zeichen in verschiedenen Farben auf den Bildschirm bringen. Die unterschiedlichen Startadressen der Grafikkarten wurden schon in der

Bild 2.
Vorsicht – Asteroid von rechts!



Initialisierungsroutine berücksichtigt. Nun wird an eine gradzahlige Adresse das Zeichen, an die folgende ungradzahlige die Farbe des Zeichens geschrieben. Ein Beispiel verdeutlicht dies:

```
10 def seg = &hb000
20 poke 0,65
30 poke 1,1
40 poke 2000,66
50 poke 2001,31
60 poke 3998,67
70 poke 3999,7
```

Zeile 10 legt die Segmentadresse für den Monochromadapter fest. Für eine CGA-Karte geben Sie hier

```
10 def seg = &hb800
ein.
```

Der erste Poke-Befehl schreibt ein »A« in die obere linke Ecke und »poke 1,1« setzt das Farbattribut auf blau. Auf einer Hercules- (also einer monochromen) Karte wird das Zeichen unterstrichen. Die beiden folgenden Anweisungen schreiben ein »B« in die Bildschirmmitte. Die Farbe ist leuchtend weiß, auf einer originalen Hercules-Karte steht das Attribut für blinkend. Die letzten beiden Be-

fehle schreiben ein »C« in normaler Farbe in die untere rechte Ecke.

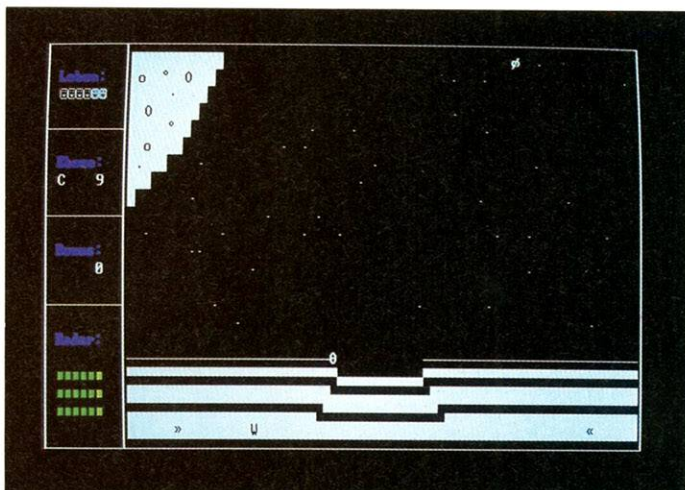
In den Zeilen 1620 bis 2190 wird das eigentliche Titelbild gezeichnet. Die zu Anfang definierten Zeichenketten werden mit Print-Anweisungen ausgegeben, alle anderen Informationen wieder mit Hilfe der schnellen Poke-Befehle.

Ab Zeile 2190 beginnt das eigentliche Spiel. Zuerst wird der Bildschirm für die Ebene A aufgebaut. Bewegungen Ihrer Gegner, das Umfärben der verschiedenen Plattformen und die Bewegung Ihrer eigenen Spielfigur finden ebenfalls hier Berücksichtigung. Ab Zeile 2820 wird das Gleiche für die Ebene B und ab Zeile 3660 für die Ebene C erledigt.

Programmenteile, auf die von verschiedenen Ebenen aus zugegriffen wird, finden Sie ab Zeile 4370. Die Punktzahl wird im Unterprogramm bis Zeile 4420 auf den Bildschirm gebracht. In der folgenden Routine wird der Bildschirm gelöscht, ohne den Maskenaufbau zu zerstören.

Etwas umständlich gestaltet sich das Zeichnen des Kastens, der immer vor dem Start einer neuen Ebene auf dem

Bild 3.
Landung verboten



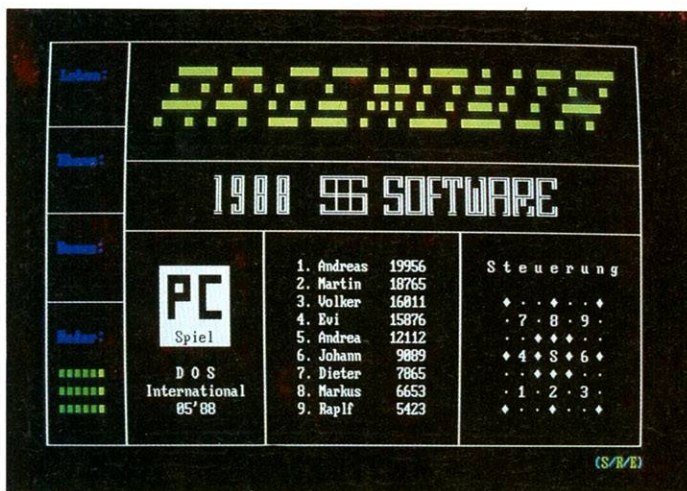


Bild 4. Der Titel zeigt die Tastensteuerung

Kasten einfärben (Ebene A):	+ 100 Punkte
Block ausleeren (Ebene B):	+ x mal 100 Punkte
Abschuß Raumschiff (Ebene C):	+ 100 Punkte
Super-Sprung (Ebene A):	- 50 Punkte
Leben verlieren:	- 100 Punkte
pro Zeiteinheit:	- 1 Punkt

Tabelle. So bekommen Sie Punkte

Bildschirm steht. Bei einem Tastendruck wird er wieder gelöscht. Die Routine zieht sich bis in die Zeile 4980, wo das Unterprogramm für das Spieleende beginnt. Sind nämlich alle Leben verbraucht, so werden diese Befehle notwendig, um eventuelle Highscore-Ergebnisse in die entsprechende Datei zu schreiben. Ein Tastendruck startet das Spiel im Anschluß daran neu.

Mit der Warteschleife in den Zeilen 5520 bis 5550 läßt sich die Spielgeschwindigkeit an Ihren Computer –

und an Ihre Fähigkeiten – anpassen. Größere Endwerte in der For...To-Anweisung verlangsamen das Programm, kleinere machen es schneller. Mit dem abgedruckten Wert »300« eignet sich das Spiel für einen mit 8 MHz getakteten PC.

Haben Sie ein Leben verwirkt, so ist die Routine ab Zeile 5560 aktiv. Das verlorene Leben wird abgezogen und – falls kein neues mehr zur Verfügung steht – der Abschnitt »Spielen-de« aufgerufen.

Die letzten zehn Unterprogramme steuern weitere Aufgaben von Argemonia. Ab Zeile 5640 und Zeile 5950 stehen jeweils Tastaturabfrageroutinen. Das »Beamen« in der Ebene A wird ab Zeile 5890 gesteuert, die Explosion in der gleichen Ebene ab Zeile 7190. Zeile 7590 ist der Start für das Fallen der Spielfigur in Ebene A. Die Unterprogramme Sonne (ab Zeile 6030) und Regeln (ab Zeile 6350) sind für die Spielbeschreibung zuständig.

Die Error-Routine ab Zeile 7710 behebt den Fehler, wenn die Datei »arge.hi« nicht auf der Diskette in Laufwerk A liegt. Haben Sie das Programm nicht korrekt abgetippt, so daß beim Programmstart auch ein anderer Fehler vorliegen kann, so wird diese Routine fälschlicherweise aufgerufen.

Falls beim Testen des Programms dieses Problem auftaucht, so ersetzen Sie die Zeile 190 durch

```
rem on error goto 7710
```

Jetzt quittiert der Interpreter jeden Fehler mit einer entsprechenden Meldung. Erst wenn das Programm einwandfrei arbeitet, entfernen Sie die Rem-Anweisung aus Zeile 190 wieder. Dann wird beim Programmablauf korrekt auf die fehlende Datei hingewiesen.

Die beiden Unterprogramme »Sterne« (ab Zeile 7780) und »Planet« (ab Zeile 7860) zeichnen die Hintergrundsterne in Ebene A und die Planetenoberfläche in den Ebenen A und C.

Die abschließenden Data-Zeilen enthalten die Werte für die verschiedenen Spielstufen. Damit erweitern sich die drei Ebenen auf insgesamt 54 verschiedene Level.

Argemonia ist in Basic geschrieben. Daß darunter aber die Struktur und die Übersichtlichkeit eines Programms keineswegs leiden muß, zeigt dieses Beispiel ganz deutlich. Die Vielzahl der Anweisungen macht es allerdings schwer, den Überblick nicht zu verlieren.

Also erschrecken Sie nicht, wenn Sie zu Anfang nicht alle Anweisungen interpretieren können. Auch wurde bei dem Programm besonderer Wert auf Geschwindigkeit gelegt, was die Klarheit einiger Befehle zugegebenermaßen beeinträchtigt.

Sie sehen aber an Argemonia, daß man auch in Basic übersichtlich und strukturiert programmieren kann.

(Arne Steingraber/hg)

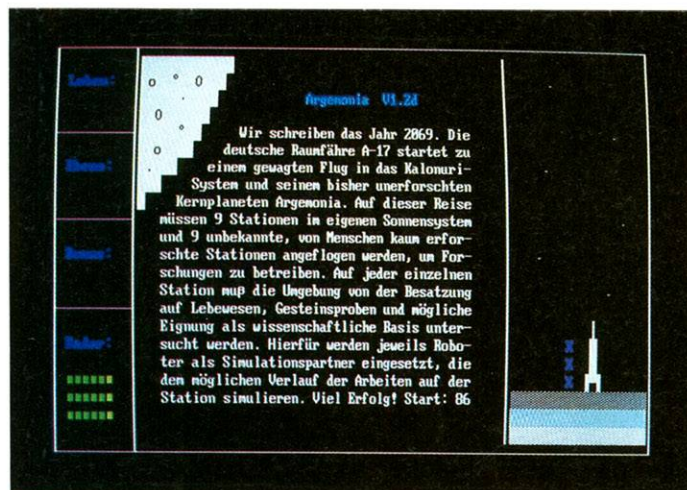


Bild 5. Das Programm erklärt sich selbst


```

[8Y1] 1560 COLOR 2,0
[UA2] 1570 LOCATE T,3
[NS3] 1580 PRINT STRING$(5,254);
[F42] 1590 COLOR 10,0
[AU2] 1600 PRINT CHR$(254)
[P91] 1610 NEXT
[PL3] 1620 REM *** TITELBILD ***
[LG2] 1630 FOR T=1 TO 4
[EL2] 1640 LOCATE T+2,12
[W2] 1650 PRINT B$(T)
[AA1] 1660 NEXT
[MY1] 1670 COLOR 7,0
[LG2] 1680 FOR T=1 TO 3
[UM2] 1690 LOCATE T+8,12
[CA2] 1700 PRINT C$(T)
[T91] 1710 NEXT
[HY1] 1720 COLOR 0,7
[6G2] 1730 FOR T=1 TO 5
[US2] 1740 LOCATE 13+T,16
[LB2] 1750 PRINT D$(T)
[DA1] 1760 NEXT
[RY1] 1770 COLOR 7,0
[5H2] 1780 FOR T=6 TO 8
[9T2] 1790 LOCATE 14+T,14
[GA2] 1800 PRINT D$(T)
[W91] 1810 NEXT
[DG2] 1820 FOR T=1 TO 9
[RS2] 1830 LOCATE 13+T,34
[0MA] 1840 PRINT CHR$(48+T);CHR$(46);CHR$(32);STRING$(9,95);CHR$(32);STRING$(5,95)
[CA1] 1850 NEXT
[BD2] 1860 LOCATE 14,60
[WV1] 1870 PRINT E$
[1H2] 1880 FOR T=1 TO 7
[PT2] 1890 LOCATE 15+T,62
[MA2] 1900 PRINT E$(T)
[Z91] 1910 NEXT
[5Z3] 1920 FOR T=495 TO 483 STEP -2
[N72] 1930 POKE T,135
[BA1] 1940 NEXT
[VE5] 1950 OPEN "A:ARGE.HI" FOR INPUT AS #1
[NL2] 1960 FOR HI=0 TO 8
[X14] 1970 INPUT #1,HNS$(HI),HP(HI)
[1W2] 1980 LOCATE 22-HI,37
[7W3] 1990 PRINT HNS$(HI);CHR$(32);
[D32] 2000 GOSUB 5520
[DC4] 2010 PRINT USING "#####";HP(HI)
[A91] 2020 NEXT
[KR1] 2030 CLOSE #1
[G72] 2040 POKE 3986,40
[P42] 2050 POKE 3987,3
[882] 2060 POKE 3988,83
[GD2] 2070 POKE 3989,138
[Y72] 2080 POKE 3990,47
[P42] 2090 POKE 3991,3
[172] 2100 POKE 3992,82
[BC2] 2110 POKE 3993,138
[N72] 2120 POKE 3994,47
[D42] 2130 POKE 3995,3
[982] 2140 POKE 3996,69
[7D2] 2150 POKE 3997,138
[582] 2160 POKE 3998,41
[952] 2170 POKE 3999,3
[V42] 2180 GOSUB 5650
[LT3] 2190 REM *** EBENE ALPHA ***
[CY1] 2200 HS="GRÜN"
[6G1] 2210 IS="*S"
[BV3] 2220 IF LV<6 THEN LV=LV+1
[YR4] 2230 IF LE<-1 THEN HP(9)=HP(9)+300
[2R1] 2240 READ LE
[742] 2250 GOSUB 4430
[AW3] 2260 FOR T=485 TO 497 STEP 2
[K72] 2270 POKE T,135
[8A1] 2280 NEXT
[182] 2290 POKE 1284,65
[E42] 2300 GOSUB 7780
[8A7] 2310 RADAR=3043+INT(ABS(LE)/6)*160+2*(ABS(LE) MOD 6)
[0N6] 2320 IF ABS(LE) MOD 6=0 THEN RADAR=RADAR-148
[RK5] 2330 POKE RADAR,PEEK(RADAR) XOR 128
[U42] 2340 GOSUB 7860
[W82] 2350 LOCATE 9,4
[TA4] 2360 PRINT USING "###";ABS(LE);
[TS2] 2370 PRINT CHR$(58)
[WM1] 2380 READ G
[SH2] 2390 FOR T=1 TO G
[R82] 2400 LOCATE 9,7
[WD3] 2410 PRINT USING "###";T
[9X1] 2420 READ X,Y
[2E2] 2430 FOR A=1 TO 3
[NF3] 2440 LOCATE X+A-1,Y-A+1
[B82] 2450 PRINT A$(A)

```

```

[6A1] 2460 NEXT
[YE2] 2470 FOR B=4 TO 6
[DG3] 2480 LOCATE X+B-1,Y-A+2
[Z82] 2490 PRINT A$(B)
[H91] 2500 NEXT
[LE2] 2510 FOR C=7 TO 8
[EH3] 2520 LOCATE X+C-1,Y-8+C
[E82] 2530 PRINT A$(C)
[1A1] 2540 NEXT
[5A1] 2550 NEXT
[RS1] 2560 READ SM
[BT1] 2570 READ ZA
[KX2] 2580 FOR AC=1 TO ZA
[G92] 2590 READ GB(AC)
[M91] 2600 NEXT
[HN2] 2610 FOR DV=1 TO G
[XW2] 2620 FOR AC=1 TO ZA
[G22] 2630 RN=RND(1)
[CRX] 2640 IF RN<.15 THEN GA=-966 ELSE IF RN>.15 AND RN<.3 THEN GA=-954 ELSE IF RN>.3 AND RN<.5 THEN GA=-12 ELSE IF RN>.5 AND RN<.65 THEN GA=12 ELSE IF RN>.65 AND RN<.85 THEN GA=954 ELSE IF RN>.85 THEN GA=966
[3ZN] 2650 IF PEEK(GB(AC)+GA)=178 OR PEEK(GB(AC)+GA)=17 OR PEEK(GB(AC)+GA)=60 OR PEEK(GB(AC)+GA)=16 OR PEEK(GB(AC)+GA)=62 THEN POKE GB(AC),178:GB(AC)=GB(AC)+GA
[UW2] 2660 POKE GB(AC),GF
[GA1] 2670 NEXT
[E8C] 2680 IF PEEK(SM)=9 OR PEEK(SM)=7 OR PEEK(SM+2)=9 OR PEEK(SM+2)=7 THEN GOSUB 7190
[W52] 2690 GOSUB 5790
[NBR] 2700 IF SA<>0 THEN POKE SM,178:POKE SM+2,178:SM=SM+SA:IF PEEK(SM)<>178 AND PEEK(SM)<>9 AND PEEK(SM)<>7 THEN GOSUB 7590 ELSE IF PEEK(SM)<>178 THEN GOSUB 7190
[R61] 2710 SA=0
[GA2] 2720 POKE SM,S1
[KC2] 2730 POKE SM+2,S2
[BF7] 2740 IF PEEK(SM-1)=7 THEN HP(9)=HP(9)+50:GOSUB 4370
[8C3] 2750 IF PEEK(SM-1)=7 THEN FOR FA=1 TO 2:POKE SM-322+FA*2+1,10:POKE SM+318+FA*2+1,10:NEXT:FOR FA=1 TO 4:POKE SM-164+FA*2+1,10:POKE SM+156+FA*2+1,10:NEXT:FOR FA=1 TO 6:POKE SM-6+FA*2+1,10:NEXT:G=G-1:LOCATE 9,7:PRINT USING "###";G:NEXT:GOTO 2820
[ME3] 2760 S1=PEEK(SM) XOR 45
[HT3] 2770 S2=PEEK(SM+2) XOR 46
[8B2] 2780 GF=GF XOR 14
[SE2] 2790 HP(9)=HP(9)-1
[F42] 2800 GOSUB 4370
[NW1] 2810 GOTO 2620
[SP3] 2820 REM *** EBENE BETA ***
[GU1] 2830 HS="GELB"
[6B1] 2840 IS="+"
[T42] 2850 GOSUB 4430
[982] 2860 POKE 1284,66
[MX1] 2870 PO(1)=182
[NX1] 2880 PO(2)=224
[5Y1] 2890 PO(3)=274
[1X1] 2900 PO(4)=316
[Z22] 2910 PO(5)=1302
[E32] 2920 PO(6)=2422
[X32] 2930 PO(7)=3542
[R42] 2940 PO(8)=3584
[G42] 2950 PO(9)=3634
[S42] 2960 PO(10)=3676
[K42] 2970 PO(11)=2556
[C42] 2980 PO(12)=1436
[V21] 2990 XY(1)=162
[0Y1] 3000 XY(2)=162
[PY1] 3010 XY(3)=158
[WY1] 3020 XY(4)=158
[D52] 3030 XY(5)=-158
[SY1] 3040 XY(6)=162
[T52] 3050 XY(7)=-158
[Z52] 3060 XY(8)=-158
[G52] 3070 XY(9)=-162
[452] 3080 XY(10)=-162
[N42] 3090 XY(11)=158
[D42] 3100 XY(12)=-162
[6B1] 3110 CH=49
[NS1] 3120 READ WU
[313] 3130 FOR ER=1 TO WU
[MF3] 3140 PX=INT(RND(1)*10)+7
[CL3] 3150 PY=INT(RND(1)*46)+23
[TB8] 3160 IF PX>8 AND PX<12 THEN IF PY>42 AND PY<46 THEN EN 3140
[RAK] 3170 IF PEEK(PX*160+PY*2)<>32 OR PEEK(PX*160+(PY+2)*2)<>32 OR PEEK((PX+2)*160+PY*2)<>32 OR PEEK((PX+2)*160+(PY+2)*2)<>32 THEN 3140
[G92] 3180 LOCATE 9,7

```



```

[BF3] 3190 PRINT USING "###";ER
[2N2] 3200 FOR TZ=0 TO 2
[UM3] 3210 LOCATE PX+TZ+1, PY+1
[FZA] 3220 IF TZ=1 THEN PRINT CHR$(186);CHR$(ER+48);CHR
$(186) ELSE PRINT WU$(TZ)
[P91] 3230 NEXT
[U91] 3240 NEXT
[V72] 3250 POKE 1294,49
[TM1] 3260 RE=1850
[961] 3270 DF=0
[XM2] 3280 FOR T=1 TO 12
[SK2] 3290 POKE PO(T),32
[6C3] 3300 PO(T)=PO(T)+XY(T)
[87G] 3310 IF PEEK(PO(T))=32 THEN 3460 ELSE IF PEEK(PO(
T))=223 OR PEEK(PO(T))=220 THEN SM=RE:POKE RE,32:
GOSUB 7220
[KC3] 3320 PO(T)=PO(T)-XY(T)
[DP1] 3330 CO=CO+1
[F06] 3340 IF CO=8 THEN CO=0:XY(T)=0:GOTO 3460
[GY3] 3350 IF XY(T)<>162 THEN 3380
[WAG] 3360 IF PEEK(PO(T)+2)<>32 THEN XY(T)=158 ELSE IF
PEEK(PO(T)+160)<>32 THEN XY(T)=-158 ELSE XY(T)=
XY(T):GOTO 3460
[XW1] 3370 GOTO 3300
[LY3] 3380 IF XY(T)<>158 THEN 3410
[MAG] 3390 IF PEEK(PO(T)-2)<>32 THEN XY(T)=162 ELSE IF
PEEK(PO(T)+160)<>32 THEN XY(T)=-162 ELSE XY(T)=
XY(T):GOTO 3460
[4W1] 3400 GOTO 3300
[E14] 3410 IF XY(T)<>-162 THEN 3440
[29G] 3420 IF PEEK(PO(T)-2)<>32 THEN XY(T)=-158 ELSE IF
PEEK(PO(T)-160)<>32 THEN XY(T)=158 ELSE XY(T)=
XY(T):GOTO 3460
[GW1] 3430 GOTO 3300
[A8G] 3440 IF PEEK(PO(T)+2)<>32 THEN XY(T)=-162 ELSE I
F PEEK(PO(T)-160)<>32 THEN XY(T)=162 ELSE XY(T)=
XY(T):GOTO 3460
[SW1] 3450 GOTO 3300
[C71] 3460 CO=0
[EG2] 3470 POKE PO(T),Z
[GA1] 3480 NEXT
[432] 3490 Z=Z XOR 4
[W62] 3500 POKE RE,32
[K72] 3510 POKE RE+1,2
[U42] 3520 GOSUB 5960
[RV1] 3530 RE=RE+DF
[TDE] 3540 IF PEEK(RE)=32 THEN 3600 ELSE IF PEEK(RE+DF)
=CH THEN 3550 ELSE SM=RE:HP(9)=HP(9)-50:GOSUB 722
0
[1V4] 3550 IF PEEK(RE+DF)=CH THEN FOR GP=CH TO 48 STEP
-1:POKE RE+DF,GP:GOSUB 5520:POKE RE-DF,43:GOSUB 5
520:POKE RE-DF,35:GOSUB 5520:POKE RE-DF,206:HP(9)
=HP(9)+50:GOSUB 4370:NEXT:POKE RE+DF,15:POKE RE+D
F+1,138:CH=CH+1:LOCATE 9,7:PRINT USING "###";CH-48
[VP3] 3560 IF CH>WU+48 THEN 3660
[FW1] 3570 RE=RE-DF
[W72] 3580 POKE RE,32
[U61] 3590 DF=0
[UC2] 3600 POKE RE,HQ
[MA2] 3610 POKE RE+1,10
[V92] 3620 HQ=HQ XOR 3
[ZD2] 3630 HP(9)=HP(9)-1
[V42] 3640 GOSUB 4370
[DX1] 3650 GOTO 3280
[LT3] 3660 REM *** EBENE GAMMA ***
[4W1] 3670 H$="BLAU"
[TH1] 3680 I$="-S"
[F52] 3690 GOSUB 4370
[W51] 3700 DF=0
[842] 3710 GOSUB 4430
[U72] 3720 POKE 1284,67
[852] 3730 GOSUB 7780
[852] 3740 GOSUB 7860
[K42] 3750 GOSUB 6030
[HY1] 3760 COLOR 1,0
[XD2] 3770 LOCATE 19,40
[UK3] 3780 PRINT STRING$(12,32)
[Z21] 3790 COLOR 7,0
[BC2] 3800 LOCATE 20,40
[6P3] 3810 PRINT STRING$(12,220)
[7D2] 3820 LOCATE 21,39
[KP3] 3830 PRINT STRING$(14,220)
[ED2] 3840 LOCATE 22,37
[986] 3850 PRINT CHR$(32);STRING$(16,223);CHR$(32)
[UC2] 3860 POKE 3555,112
[GD2] 3870 POKE 3554,175
[5D2] 3880 POKE 3665,112
[SD2] 3890 POKE 3664,174
[LT2] 3900 FOR CV=1 TO 10
[C92] 3910 LOCATE 9,7
[GZ3] 3920 PRINT USING "###";10-CV
[PL6] 3930 IF RND(1)>.5 THEN QR=2:QS=2982:GOTO 3950

```

```

[2L2] 3940 QR=-2:QS=2958
[XK4] 3950 FOR EP=286 TO 2840 STEP 158
[T12] 3960 X$=INKEY$
[UP2] 3970 X$=RIGHT$(X$,1)
[2D5] 3980 IF US THEN GOSUB 4300:GOTO 4000
[ZFE] 3990 IF X$=CHR$(83) OR X$=CHR$(115) THEN US=1:SQ=
AL:IF AL>3554 AND AL<3592 THEN TS=-158 ELSE TS=-1
62
[232] 4000 GOSUB 6000
[2F6] 4010 IF DF<>0 THEN POKE AL,32:AL=AL+DF:DF=0
[GKG] 4020 IF AL=3554 THEN AL=3662 ELSE IF AL=3664 THEN
AL=3556 ELSE IF AL=3590 THEN AL=3630 ELSE IF AL=
3628 THEN AL=3588
[7D2] 4030 POKE AL+1,112
[562] 4040 POKE AL,87
[X15] 4050 IF LD THEN GOSUB 5520:RETURN
[VD2] 4060 HP(9)=HP(9)-1
[R42] 4070 GOSUB 4370
[002] 4080 QS=QS+QR
[H92] 4090 POKE QS,233
[1F2] 4100 EU=PEEK(EP)
[572] 4110 POKE EP,237
[W32] 4120 GOSUB 5520
[M92] 4130 POKE QS,196
[VB2] 4140 POKE EP,EU
[X91] 4150 NEXT
[P61] 4160 LD=1
[ZC2] 4170 POKE 2972,248
[652] 4180 GOSUB 3960
[W72] 4190 POKE 2972,32
[3C2] 4200 POKE 2970,249
[D42] 4210 GOSUB 3960
[X62] 4220 POKE 2970,32
[082] 4230 POKE 2968,46
[T42] 4240 GOSUB 3960
[Z72] 4250 POKE 2968,32
[R61] 4260 LD=0
[8A1] 4270 NEXT
[CL5] 4280 POKE RADAR,PEEK(RADAR) XOR 128
[VM5] 4290 IF LE>-18 THEN 2190 ELSE GOTO 4980
[1D2] 4300 POKE SQ,JO
[EZ1] 4310 SQ=SQ+TS
[MG2] 4320 JO=PEEK(SQ)
[A82] 4330 POKE SQ,19
[MN3] 4340 IF SQ=QS OR SQ=QS+QR THEN FOR T=1 TO 5:DL1=S
Q-(T*162):JO1=PEEK(DL1):POKE DL1,219:DL2=SQ-(T*16
0):JO2=PEEK(DL2):POKE DL2,219:DL3=SQ-(T*158):JO3=
PEEK(DL3):POKE DL3,219:GOSUB 5520:POKE DL1,JO1:PO
KE DL2,JO2:POKE DL3,JO3:NEXT:POKE SQ,196
[7U9] 4350 IF SQ=QS OR SQ=QS+QR THEN US=0:HP(9)=HP(9)+1
00:SQ=0:RETURN
[F68] 4360 IF SQ+TS<2900 THEN US=0:POKE SQ,196:SQ=0:RET
URN
[6A3] 4370 REM *** PUNKTE ***
[P04] 4380 IF HP(9)<0 THEN HP(9)=0
[485] 4390 IF HP(9)>999991 THEN HP(9)=999991
[B92] 4400 LOCATE 14,4
[V34] 4410 PRINT USING "#####";HP(9)
[KV1] 4420 RETURN
[HL5] 4430 REM *** BILDSCHIRM LÖSCHEN ***
[PM2] 4440 FOR T=2 TO 12
[ZB2] 4450 LOCATE T,12
[E03] 4460 PRINT SPACE$(68)
[3T2] 4470 LOCATE 25-T,12
[P03] 4480 PRINT SPACE$(68)
[PA1] 4490 NEXT
[3C4] 4500 IF OF THEN OF=0:RETURN
[763] 4510 REM *** BANNER ***
[M42] 4520 GOSUB 5560
[8N6] 4530 F$(2)=F1$(2)+G$(ABS(LE))+ " Phase: "+H$+F2$(
2)
[C38] 4540 F$(3)=F1$(3)+"Steuerung: "+I$+" Start: <-|
"+F2$(3)
[MR2] 4550 FOR BA=1 TO 32
[YC2] 4560 LOCATE 11,12
[LX3] 4570 PRINT RIGHT$(F$(1),BA)
[WV2] 4580 LOCATE 12,80-BA
[SN3] 4590 PRINT LEFT$(F$(2),BA)
[FC2] 4600 LOCATE 13,12
[4X3] 4610 PRINT RIGHT$(F$(3),BA)
[DV2] 4620 LOCATE 14,80-BA
[7N3] 4630 PRINT LEFT$(F$(4),BA)
[8A1] 4640 NEXT
[9S2] 4650 FOR BA=1 TO 18
[BV2] 4660 LOCATE 11,12+BA
[272] 4670 PRINT F$(1)

```

Listing 2. Action gibt es auch in Basic


```

[DW2] 4680 LOCATE 12,49-BA
[E72] 4690 PRINT F$(2)
[WU2] 4700 LOCATE 13,12+BA
[N62] 4710 PRINT F$(3)
[YV2] 4720 LOCATE 14,49-BA
[172] 4730 PRINT F$(4)
[BA1] 4740 NEXT
[XY4] 4750 IF INKEY$<>CHR$(13) THEN 4750
[8K2] 4760 FOR BA=1 TO 9
[8N2] 4770 FOR WL=1 TO 4
[TC3] 4780 LOCATE 1+BA,11+BA*2
[DY2] 4790 PRINT CHR$(SX)
[GD3] 4800 LOCATE 1+BA,79-BA*2
[NW2] 4810 PRINT CHR$(SO)
[WF3] 4820 LOCATE 24-BA,11+BA*2
[XW2] 4830 PRINT CHR$(SO)
[8H3] 4840 LOCATE 24-BA,79-BA*2
[ØY2] 4850 PRINT CHR$(SX)
[TN2] 4860 SX=SX XOR 124
[8F2] 4870 SO=SO XOR 15
[YA1] 4880 NEXT
[2B1] 4890 NEXT
[7H2] 4900 FOR BA=1 TO 4
[914] 4910 TG$=STRING$(30,TG(BA))
[BM2] 4920 FOR FU=1 TO 4
[LV2] 4930 LOCATE 10+FU,31
[U12] 4940 PRINT TG$
[NA1] 4950 NEXT
[TA1] 4960 NEXT
[NW1] 4970 RETURN
[FN3] 4980 REM *** SPIELENDEN ***
[B81] 4990 OF=1
[L32] 5000 GOSUB 4430
[M92] 5010 LOCATE 6,35
[NN3] 5020 PRINT STRING$(21,220)
[Y92] 5030 LOCATE 7,35
[KWA] 5040 PRINT"■*S*p*i*e*1*e*n*d*e*■"
[8A2] 5050 LOCATE 8,35
[BP3] 5060 PRINT STRING$(21,223)
[U44] 5070 IF HP(9)<=HP(Ø) THEN 5480
[ZC2] 5080 LOCATE 11,34
[842] 5090 COLOR 10,Ø
[CW5] 5100 PRINT"Herzlichen Glückwunsch!"
[7C2] 5110 LOCATE 13,22
[Z32] 5120 COLOR 18,Ø
[HC9] 5130 PRINT"Bitte Namen für die Highscore-Tabelle
    eingeben"
[1Y1] 5140 COLOR 7,Ø
[ØD2] 5150 LOCATE 16,41
[MG3] 5160 PRINT STRING$(9,95)
[181] 5170 CT=1
[U53] 5180 X=INT(RND(1)*22)+2
[483] 5190 Y=INT(RND(1)*66)+13
[G28] 5200 IF SCREEN(X,Y)=32 THEN LOCATE X,Y:PRINT CHR$(
    250)
[RF2] 5210 FOR T=1 TO 5
[553] 5220 X=INT(RND(1)*22)+2
[F73] 5230 Y=INT(RND(1)*66)+13
[TD8] 5240 IF SCREEN(X,Y)=250 THEN LOCATE X,Y:PRINT CH
    R$(32)
[2A1] 5250 NEXT
[6W2] 5260 LOCATE 16,40+CT
[KL4] 5270 X$=INKEY$:IF X$="" THEN 5180
[3R8] 5280 IF X$=CHR$(13) THEN HN$(9)=HN$(9)+SPACE$(10-
    CT):GOTO 5370
[PX7] 5290 IF ASC(X$)>64 AND ASC(X$)<91 THEN CT=CT+1:GO
    TO 5330
[W27] 5300 IF ASC(X$)>96 AND ASC(X$)<123 THEN CT=CT+1:G
    OTO 5330
[B27] 5310 IF ASC(X$)=32 OR ASC(X$)=45 THEN CT=CT+1:GOT
    O 5330
[2X1] 5320 GOTO 5270
[SV1] 5330 PRINT X$
[YN2] 5340 HN$(9)=HN$(9)+X$
[DA3] 5350 IF CT=10 THEN 5370
[LX1] 5360 GOTO 5180
[RG2] 5370 FOR T=Ø TO 8
[GP2] 5380 FOR R=T+1 TO 9
[ØA4] 5390 IF HP(T)<=HP(R) THEN 5410
[W4A] 5400 PK=HP(T):HP(T)=HP(R):HP(R)=PK:HK$=HN$(T):HN$(
    T)=HN$(R):HN$(R)=HK$
[S91] 5410 NEXT
[W91] 5420 NEXT
[4R5] 5430 OPEN "A:ARGE.HI" FOR OUTPUT AS #1
[HG2] 5440 FOR T=1 TO 9
[2U3] 5450 WRITE #1,HN$(T),HP(T)
[CA1] 5460 NEXT
[MS1] 5470 CLOSE #1
[CD2] 5480 LOCATE 13,22
[MV9] 5490 PRINT "      Zur Fortsetzung beliebige Taste d
    rücken "
[CW3] 5500 IF INKEY$="" THEN 5500

```

```

[N61] 5510 RUN
[HB4] 5520 REM *** WARTESCHLEIFE ***
[9X2] 5530 FOR QW=1 TO 150
[7A1] 5540 NEXT
[3W1] 5550 RETURN
[2X2] 5560 REM *** LEBEN ***
[8U2] 5570 FOR T=1 TO LV
[ØR2] 5580 POKE 496-T*2,2
[VA1] 5590 NEXT
[MV2] 5600 FOR T=LV+1 TO 6
[3P2] 5610 POKE 496-T*2,1
[2A1] 5620 NEXT
[YV1] 5630 RETURN
[L24] 5640 REM *** TASTATURABFRAGEN ****
[G12] 5650 X$=INKEY$
[1VB] 5660 IF X$<>"r" AND X$<>"R" AND X$<>"s" AND X$<>"
    S" AND X$<>"e" AND X$<>"E" THEN 5650
[ØD2] 5670 POKE 1140,179
[KD2] 5680 POKE 1780,179
[7E2] 5690 POKE 1278,179
[1D2] 5700 POKE 1918,179
[4D2] 5710 POKE 3736,196
[2D2] 5720 POKE 3790,196
[1B4] 5730 FOR T=3986 TO 3998 STEP 2
[812] 5740 POKE T,32
[HA1] 5750 NEXT
[YC6] 5760 IF X$="R" OR X$="r" THEN OF=1:GOTO 6250
[DYL] 5770 IF X$="E" OR X$="e" THEN CLS:PRINT TAB(22);"
    Hiermit verabschiedet sich ARGEMONIA":FOR T=1 TO
    10:GOSUB 5520:NEXT:CLS:SYSTEM
[NW1] 5780 RETURN
[322] 5790 X$=INKEY$
[ZN2] 5800 X$=RIGHT$(X$,1)
[YG4] 5810 IF X$=CHR$(71) THEN SA=-966
[2H4] 5820 IF X$=CHR$(73) THEN SA=-954
[ME4] 5830 IF X$=CHR$(75) THEN SA=-12
[E57] 5840 IF X$=CHR$(83) OR X$=CHR$(115) THEN GOSUB 58
    90
[594] 5850 IF X$=CHR$(77) THEN SA=12
[SG4] 5860 IF X$=CHR$(79) THEN SA=954
[HG4] 5870 IF X$=CHR$(81) THEN SA=966
[SW1] 5880 RETURN
[373] 5890 REM *** BEAMEN ***
[222] 5900 SI=RND(1)
[42C] 5910 IF SI<.3 THEN ST=656 ELSE IF SI>.3 AND SI<.7
    THEN ST=1622 ELSE IF SI>.7 THEN ST=2576
[TH4] 5920 ST=ST+(INT(RND(1)*11)+1)*12
[6NY] 5930 IF PEEK(ST)=178 AND PEEK(ST+2)=178 THEN POKE
    SM,UN1:POKE SM+2,UN1:UN1=178:UN2=178:SM=ST:HP(9)
    =HP(9)-50:FOR T=1 TO 30:X$=INKEY$:NEXT:POKE SM,18
    5:POKE SM+2,204:FOR T=1 TO 15:GOSUB 5520:NEXT:RET
    URN
[6Y1] 5940 GOTO 5890
[TA5] 5950 REM *** TASTATURABFRAGEN II ***
[X12] 5960 X$=INKEY$
[YP2] 5970 X$=RIGHT$(X$,1)
[9G4] 5980 IF X$=CHR$(72) THEN DF=-160
[SE4] 5990 IF X$=CHR$(80) THEN DF=160
[M64] 6000 IF X$=CHR$(75) THEN DF=-2
[K24] 6010 IF X$=CHR$(77) THEN DF=2
[AV1] 6020 RETURN
[9Y2] 6030 REM *** SONNE ***
[EY1] 6040 COLOR Ø,7
[L92] 6050 LOCATE 2,12
[C22] 6060 PRINT SPACE$(12)
[X92] 6070 LOCATE 3,12
[L8B] 6080 PRINT CHR$(32);CHR$(111);SPACE$(2);CHR$(248)
    ;SPACE$(2);CHR$(79);SPACE$(3)
[7A2] 6090 LOCATE 4,12
[KM5] 6100 PRINT SPACE$(5);CHR$(249);SPACE$(4)
[C92] 6110 LOCATE 5,12
[6P5] 6120 PRINT SPACE$(2);CHR$(79);SPACE$(6)
[P92] 6130 LOCATE 6,12
[SM5] 6140 PRINT SPACE$(5);CHR$(248);SPACE$(2)
[ØA2] 6150 LOCATE 7,12
[7M5] 6160 PRINT SPACE$(2);CHR$(111);SPACE$(4)
[AA2] 6170 LOCATE 8,12
[TE5] 6180 PRINT CHR$(32);CHR$(46);SPACE$(3)
[MA2] 6190 LOCATE 9,12
[7W2] 6200 PRINT SPACE$(3)
[2C2] 6210 LOCATE 10,12
[MR2] 6220 PRINT CHR$(32)
[2Y1] 6230 COLOR 7,Ø
[SV1] 6240 RETURN
[Y73] 6250 REM *** REGELN ***
[L42] 6260 GOSUB 4430
[H42] 6270 GOSUB 6030
[EA4] 6280 FOR T=280 TO 3640 STEP 160
[Y82] 6290 POKE T,179
[L91] 6300 NEXT
[FC2] 6310 LOCATE 21,62
[UP3] 6320 PRINT STRING$(18,176)

```



```

[UC2] 6330 LOCATE 22,62
[4R3] 6340 PRINT STRING$(18,177)
[5D2] 6350 LOCATE 23,62
[ER3] 6360 PRINT STRING$(18,178)
[W72] 6370 POKE 452,250
[Z82] 6380 POKE 948,250
[6C2] 6390 POKE 1250,250
[AB2] 6400 POKE 1750,250
[1C2] 6410 POKE 2048,250
[AC2] 6420 POKE 2386,250
[NH4] 6430 FOR T=2856 TO 3176 STEP 160
[X12] 6440 POKE T,88
[K62] 6450 POKE T+1,1
[EA1] 6460 NEXT
[4D2] 6470 POKE 2702,179
[AD2] 6480 POKE 2862,219
[2D2] 6490 POKE 3022,219
[NB2] 6500 POKE 3180,222
[2C2] 6510 POKE 3182,223
[6C2] 6520 POKE 3184,221
[8A2] 6530 LOCATE 4,34
[HY1] 6540 COLOR 9,0
[5A4] 6550 PRINT"Argemonia V1.2d"
[RY1] 6560 COLOR 7,0
[TA2] 6570 LOCATE 6,25
[VU6] 6580 PRINT "Wir schreiben das Jahr 2069. Die"
[1B2] 6590 LOCATE 7,23
[XF7] 6600 PRINT"deutsche Raumfähre A-17 startet zu"
[4A2] 6610 LOCATE 8,21
[4K7] 6620 PRINT"einem gewagten Flug in das Kalonuri-"
[KA2] 6630 LOCATE 9,19
[VD8] 6640 PRINT"System und seinem bisher unerforschten"

[8D2] 6650 LOCATE 10,17
[3H8] 6660 PRINT"Kernplaneten Argemonia. Auf dieser Rei
se"
[FD2] 6670 LOCATE 11,15
[P59] 6680 PRINT"müssen 9 Stationen im eigenen Sonnensy
stem"
[UD2] 6690 LOCATE 12,15
[KK8] 6700 PRINT"und 9 unbekannte, von Menschen kaum er
for-"
[0D2] 6710 LOCATE 13,15
[NS8] 6720 PRINT"schte Stationen angefliegen werden, um
For-"
[BD2] 6730 LOCATE 14,15
[DU8] 6740 PRINT"sungen zu betreiben. Auf jeder einze
lne"
[PD2] 6750 LOCATE 15,15
[YA9] 6760 PRINT"Station muß die Umgebung von der Besat
zung"
[1E2] 6770 LOCATE 16,15
[M39] 6780 PRINT"auf Lebewesen, Gesteinsproben und mögl
iche"
[CE2] 6790 LOCATE 17,15
[8R8] 6800 PRINT"Eignung als wissenschaftliche Basis un
ter-"
[KD2] 6810 LOCATE 18,15
[899] 6820 PRINT"sucht werden. Hierfür werden jeweils R
obo-"
[XD2] 6830 LOCATE 19,15
[7X8] 6840 PRINT"ter als Simulationspartner eingesetzt,
die"
[CD2] 6850 LOCATE 20,15
[NR8] 6860 PRINT"dem möglichen Verlauf der Arbeiten auf
der"
[RD2] 6870 LOCATE 21,15
[T98] 6880 PRINT"Station simulieren. Viel Erfolg! Start
:"
[9S3] 6890 FOR T=99 TO 0 STEP -1
[0D2] 6900 LOCATE 21,54
[UB3] 6910 PRINT USING "###";T
[S42] 6920 GOSUB 5520
[W42] 6930 GOSUB 5520
[NA1] 6940 NEXT
[NN2] 6950 FOR T=1 TO 15
[K33] 6960 POKE 2862-T*160,32
[A33] 6970 POKE 3022-T*160,32
[V33] 6980 POKE 3182-T*160,32
[L33] 6990 POKE 3340-T*160,32
[W13] 7000 POKE 3342-T*160,32
[623] 7010 POKE 3344-T*160,32
[UL3] 7020 POKE 2862-(T+1)*160,179
[XK3] 7030 POKE 3022-(T+1)*160,219
[EL3] 7040 POKE 3182-(T+1)*160,219
[3L3] 7050 POKE 3340-(T+1)*160,222
[EL3] 7060 POKE 3342-(T+1)*160,223

```

Listing 2. Action gibt es auch in Basic

Meisterstücke der Computergrafik

Der FRAKTAL-GENERATOR entführt Sie in märchenhafte Landschaften voller bizarrer Formen und Farben. Werden Sie zum Entdecker im Wunderland der Fraktale!

Die höchst-effektive Programmierung macht die Fraktale..

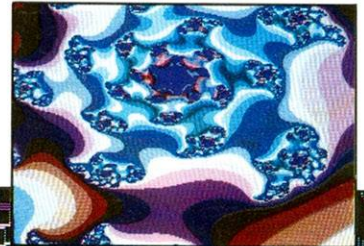
..unglaublich schnell: ob zweidimensional oder dreidimensional, ein Bild ist auf einem 12MHz-AT in weniger als 50 Sekunden fertig. High-Speed-Tuning macht Ihren Amiga zum Superrechner: **nur noch 7 Sekunden** für das "Apfelmännchen"!!

..komfortabel: Bedienung über Pulldown-Menüs, wahlweise mit Maus oder Tastatur. Hardcopies. Mehrere separate Bildspeicher. Der Amiga speichert im IFF-Format ab. Die Bilder lassen sich dadurch in andere Malprogramme laden.

..farbenfroh: Separate Farbuordnung und Animationsmöglichkeit, und die Farben lassen sich auch nachträglich beliebig verändern.

3D-Version: Mandelbrot- und Juliamenge. Blickwinkel veränderbar. Betrachten Sie das "Fraktalobjekt" von allen Seiten. Höhenfaktor. Mit Glättungsfunktion und Berg- oder Taldarstellung.

So sieht die Landschaft
in der 'alten', zweidimen-
sionalen Darstellung aus.



Ein dreidimensionaler Ausschnitt des Apfelmännchens ergibt eine futuristische Landschaft.
Autor: Matthias Uphoff

System:

Commodore Amiga mit 512 KB oder MS-DOS ab 2.0, PC-XT/AT mit EGA-Karte oder Amstrad/Schneider PC 1512.

FRAKTAL
Generator 2D

für PC

MS-DOS 5 1/4"-Disk. Best.-Nr. 225 MS-DOS 3 1/2"-Disk. Best.-Nr. 231

49,- DM (unverbindliche Preisempfehlung)			
Wenn Sie über den DMV-Bestellservice bestellen, gilt folgendes:			
Inland:		Ausland:	
Einzelpreis	49,- DM	Einzelpreis	49,- DM
zzgl. Versandkosten	3,- DM	zzgl. Versandkosten	5,- DM
Endpreis	52,- DM	Endpreis	54,- DM

FRAKTAL
Generator 3D

für PC u. Amiga

MS-DOS 5 1/4"-Disk. Best.-Nr. 248 MS-DOS 3 1/2"-Disk. Best.-Nr. 249
Amiga 3 1/2" Best.-Nr. 2901

69,- DM (unverbindliche Preisempfehlung)			
Wenn Sie über den DMV-Bestellservice bestellen, gilt folgendes:			
Inland:		Ausland:	
Einzelpreis	69,- DM	Einzelpreis	69,- DM
zzgl. Versandkosten	3,- DM	zzgl. Versandkosten	5,- DM
Endpreis	72,- DM	Endpreis	74,- DM

Bitte benutzen Sie die Bestellkarte


```

[PL3] 7070 POKE 3344-(T+1)*160,221
[3C3] 7080 FOR R=1 TO (15-T)*3
[U43] 7090 POKE 3342-T*160,239
[Z13] 7100 POKE 3342-T*160,32
[L91] 7110 NEXT
[R91] 7120 NEXT
[X24] 7130 FOR T=300 TO 780 STEP 160
[U02] 7140 POKE T,32
[782] 7150 POKE T+2,32
[F82] 7160 POKE T+4,32
[BA1] 7170 NEXT
[7W1] 7180 RETURN
[TS3] 7190 REM *** EXPLOSION ***
[D92] 7200 POKE SM,178
[NG2] 7210 POKE SM+2,178
[KM2] 7220 FOR PS=1 TO 4
[P32] 7230 EN(PS)=SM
[RZ1] 7240 ET(PS)=10
[B52] 7250 BH(PS)=219
[AA1] 7260 NEXT
[132] 7270 EV(1)=-160
[442] 7280 EV(2)=-158
[EN1] 7290 EV(3)=2
[GW1] 7300 EV(4)=162
[GW1] 7310 EV(5)=160
[CX1] 7320 EV(6)=158
[0S1] 7330 EV(7)=-2
[E32] 7340 EV(8)=-162
[BU2] 7350 FOR EX=1 TO 10
[8K2] 7360 FOR EA=1 TO 8
[FH4] 7370 EO(EA)=PEEK(EN(EA)+EV(EA))
[YP4] 7380 EW(EA)=PEEK(EN(EA)+EV(EA)+1)
[UGU] 7390 IF EO(EA)=196 OR EO(EA)=179 OR EO(EA)=191 OR
EO(EA)=217 OR EO(EA)=180 OR EO(EA)=193 OR EO(EA)=
194 THEN EO(EA)=PEEK(EN(EA)):EW(EA)=PEEK(EN(EA)+
1):EV(EA)=0:BH(EA)=EO(EA):ET(EA)=EW(EA)
[S91] 7400 NEXT
[RH2] 7410 FOR EA=1 TO 8
[HD3] 7420 EN(EA)=EN(EA)+EV(EA)
[CD3] 7430 POKE EN(EA),BH(EA)
[YF3] 7440 POKE EN(EA)+1,ET(EA)
[CA1] 7450 NEXT
[6P3] 7460 FOR T=1 TO 25:NEXT
[FK2] 7470 FOR EA=1 TO 8
[5F3] 7480 POKE EN(EA),EO(EA)
[SG3] 7490 POKE EN(EA)+1,EW(EA)
[V91] 7500 NEXT
[Z91] 7510 NEXT
[XT1] 7520 LV=LV-1
[LB3] 7530 IF LV=-1 THEN 4980
[MM2] 7540 HP(9)=HP(9)-50
[452] 7550 GOSUB 4370
[8A5] 7560 IF PEEK(1284)=65 THEN GOSUB 5890
[E52] 7570 GOSUB 5560
[LW1] 7580 RETURN
[8P2] 7590 REM *** FALL ***
[XW4] 7600 FOR FL=1 TO INT((3516-SM)/160)
[2Y4] 7610 UN1=PEEK(SM):UN2=PEEK(SM+2)
[DC4] 7620 POKE SM,S1:POKE SM+2,S2
[952] 7630 GOSUB 5960
[T42] 7640 GOSUB 5520
[ZM4] 7650 POKE SM,UN1:POKE SM+2,UN2
[812] 7660 SM=SM+160
[UA1] 7670 NEXT
[YY4] 7680 UN1=PEEK(SM):UN2=PEEK(SM+2)
[952] 7690 GOSUB 7220
[TV1] 7700 RETURN
[Z22] 7710 REM *** ERROR ***
[R41] 7720 CLS
[D42] 7730 COLOR 31,0
[Y82] 7740 LOCATE 3,3
[54F] 7750 PRINT"Bitte Diskette mit Datei »ARGE.HI« in
Laufwerk A: einlegen und Taste drücken"
[6Y3] 7760 IF INKEY$="" THEN 7760
[XF5] 7770 IF HP(9) THEN GOTO 5430 ELSE RUN
[MA3] 7780 REM *** STERNE ***
[UN2] 7790 FOR T=1 TO 50
[L53] 7800 X=INT(RND(1)*22)+2
[X73] 7810 Y=INT(RND(1)*66)+13
[Vf2] 7820 LOCATE X,Y
[SU2] 7830 PRINT CHR$(250)
[MA1] 7840 NEXT
[GW1] 7850 RETURN
[V93] 7860 REM *** PLANET ***
[9E2] 7870 LOCATE 19,12
[AS3] 7880 PRINT STRING$(68,196)
[SD2] 7890 LOCATE 20,12
[4R3] 7900 PRINT STRING$(68,223)
[YC2] 7910 LOCATE 21,12
[RR3] 7920 PRINT STRING$(68,219)
[9D2] 7930 LOCATE 22,12
[ER3] 7940 PRINT STRING$(68,220)

```

```

[MD2] 7950 LOCATE 23,12
[7S3] 7960 PRINT STRING$(68,219)
[VW1] 7970 RETURN
[473] 7980 REM *** LEVEL 01 ***
[GFG] 7990 DATA -01,18,3,45,9,18,9,30,9,42,9,48,9,60,9,
72,15,15,15,21,15,27,15,33,15,39,15,45,15,51,15,5
7,15,63,15,69,15,75,728,2,2588,2710,6
[D53] 8000 REM *** LEVEL 02 ***
[63C] 8010 DATA -02,12,3,39,3,45,3,51,9,36,9,42,9,48,9,
54,15,33,15,39,15,45,15,51,15,57,728,2,2624,2674,
8
[R53] 8020 REM *** LEVEL 03 ***
[BPH] 8030 DATA -03,20,3,15,3,33,3,39,3,45,3,51,3,57,3,
75,9,18,9,30,9,42,9,48,9,60,9,72,15,21,15,27,15,3
9,15,45,15,51,15,63,15,69,728,3,2636,2650,2662,7
[163] 8040 REM *** LEVEL 04 ***
[C6H] 8050 DATA -04,20,3,21,3,27,3,39,3,45,3,51,3,63,3,
69,9,18,9,30,9,36,9,54,9,60,9,72,15,21,15,27,15,3
9,15,45,15,51,15,63,15,69,2648,2,1634,1744,3
[B63] 8060 REM *** LEVEL 05 ***
[6VK] 8070 DATA -05,22,3,15,3,21,3,27,3,45,3,63,3,69,3,
75,9,18,9,24,9,42,9,48,9,66,9,72,15,21,15,27,15,3
3,15,39,15,45,15,51,15,57,15,63,15,69,2648,3,692,
730,766,5
[N63] 8080 REM *** LEVEL 06 ***
[HXG] 8090 DATA -06,20,3,15,3,33,3,39,3,45,3,51,3,57,3,
75,9,18,9,24,9,30,9,60,9,66,9,72,15,15,15,33,15,3
9,15,45,15,51,15,57,15,75,2648,2,668,790,4
[U53] 8100 REM *** LEVEL 07 ***
[L7F] 8110 DATA -07,18,3,33,3,39,3,51,3,57,3,63,3,69,3,
75,9,30,9,42,9,48,9,60,15,15,15,21,15,27,15,33,15
,39,15,51,15,57,2588,1,788,9
[463] 8120 REM *** LEVEL 08 ***
[MGN] 8130 DATA -08,26,3,15,3,21,3,27,3,33,3,39,3,45,3,
51,3,57,3,63,3,69,3,75,9,18,9,42,9,48,9,72,15,15,
15,21,15,27,15,33,15,39,15,45,15,51,15,57,15,63,1
5,69,15,75,2648,3,668,730,790,7
[E63] 8140 REM *** LEVEL 09 ***
[56F] 8150 DATA -09,18,3,15,3,21,3,39,3,51,3,57,3,69,9,
24,9,30,9,42,9,48,9,60,9,66,15,21,15,33,15,39,15,
51,15,69,15,75,2708,1,668,4
[563] 8160 REM *** LEVEL 10 ***
[BZG] 8170 DATA -10,20,3,21,3,39,3,57,3,63,3,69,3,75,9,
18,9,24,9,36,9,42,9,48,9,54,9,66,9,72,15,15,15,21
,15,27,15,33,15,51,15,69,680,2,2660,2698,8
[F63] 8180 REM *** LEVEL 11 ***
[PFK] 8190 DATA -11,22,3,15,3,21,3,27,3,33,3,39,3,57,3,
63,3,75,9,18,9,30,9,42,9,48,9,60,9,72,15,15,15,27
,15,33,15,51,15,57,15,63,15,69,15,75,1634,2,1718,
1744,5
[M53] 8200 REM *** LEVEL 12 ***
[01K] 8210 DATA -12,22,3,15,3,21,3,33,3,45,3,57,3,69,3,
75,9,24,9,30,9,36,9,42,9,48,9,54,9,60,9,66,15,15,
15,21,15,33,15,45,15,57,15,69,15,75,728,2,668,790
,3
[Y53] 8220 REM *** LEVEL 13 ***
[ZAK] 8230 DATA -13,22,3,15,3,21,3,33,3,39,3,45,3,51,3,
63,3,69,9,18,9,30,9,42,9,48,9,60,9,72,15,21,15,27
,15,39,15,45,15,51,15,57,15,69,15,75,2708,2,668,6
82,6
[863] 8240 REM *** LEVEL 14 ***
[AHH] 8250 DATA -14,21,3,15,3,21,3,27,3,39,3,45,3,51,3,
63,3,69,3,75,9,18,9,24,9,36,9,54,9,66,9,72,15,15,
15,27,15,33,15,57,15,63,15,75,728,2,2588,2710,9
[K63] 8260 REM *** LEVEL 15 ***
[4S8] 8270 DATA -15,8,3,39,3,45,3,51,9,36,9,54,15,39,15
,45,15,51,728,1,2648,4
[W63] 8280 REM *** LEVEL 16 ***
[YBK] 8290 DATA -16,22,3,15,3,21,3,27,3,45,3,63,3,69,3,
75,9,18,9,30,9,36,9,42,9,48,9,54,9,60,9,72,15,15,
15,21,15,27,15,45,15,63,15,69,15,75,2648,2,1634,1
744,7
[163] 8300 REM *** LEVEL 17 ***
[Z7G] 8310 DATA -17,18,3,39,3,45,3,51,9,30,9,36,9,42,9,
48,9,54,9,60,15,21,15,27,15,33,15,39,15,45,15,51,
15,57,15,63,15,69,2648,2,2600,2698,5
[B63] 8320 REM *** LEVEL 18 ***
[KNL] 8330 DATA -18,24,3,15,3,21,3,27,3,39,3,45,3,51,3,
69,3,75,9,18,9,30,9,36,9,42,9,48,9,54,9,60,9,72,1
5,15,15,21,15,39,15,45,15,51,15,63,15,69,15,75,72
8,2,1634,1744,9
[BP2] 8340 REM *** ENDE ***

```

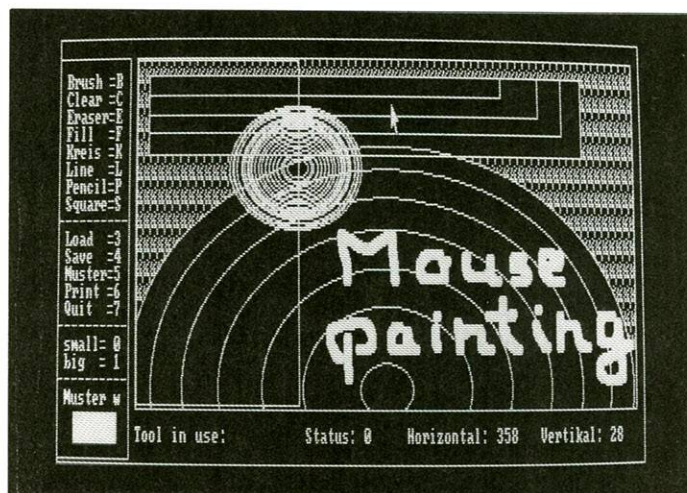
Listing 2. Action gibt es auch in Basic

Mouse-Painting – ein Malprogramm mit professionellem Touch

Dieses Basic-Programm realisiert ein professionell anmutendes Zeichenprogramm auf kompatiblen Computern mit CGA-Karte und Maus. Dabei kann sich die Arbeitsgeschwindigkeit sehen lassen.

Geschickte Programmierung und die Verwendung von Treiber Routinen verleihen diesem Programm eine für Basic überraschende Geschwindigkeit. Die sehr übersichtliche Benutzerführung und die einfache Handhabung machen es zu einer Software, die für sich selber spricht. Das Programm eignet sich hervorragend für die Überarbeitung bereits bestehender Grafiken, aber auch das Anfertigen kleinerer Bilder macht Freude. Voraussetzung ist ein installierter Maustreiber, ansonsten genügt der Aufruf »basic mzeich«.

Die Bedienung ist derartig einfach, daß es kaum weitergehender Erklärungen zur Handhabung bedarf.



Die Benutzeroberfläche von Mouse-Paint

Der Bildschirm ist in drei Bereiche unterteilt:

- das Zeichenblatt, in dem nur die Maus Anwendung findet
- die Menüleiste am linken Rand zur Auswahl der Zeichengeräte, Muster und der Lade- und Speicherroutinen
- der untere Bildschirmkasten mit der Angabe der augenblicklich angewählten Funktion, der Koordinaten und der Programmierungen.

Nach dem Programmstart wählt man mit der Tastatur ein Zeichengerät aus, das dann unter »Tool« angezeigt wird. Ab diesem Zeitpunkt ist dann das Zeichengerät aktiv. Sie können nun sofort mit der linken Maustaste mit dem Zeichnen beginnen; mit der rechten Maustaste legen Sie das Zeichengerät wieder ab.

Beim Pinsel und beim Radierer wird zwischen zwei Größen unterschieden. Der große und kleine Pinsel wird jeweils mit »1« und »0« angewählt. Die Tabelle zeigt alle verfügbaren Funktionen.

(Uwe Bergmann/db)

b	Brush:	Pinsel in zwei Größen.
c	Clear:	Bildschirm vollständig löschen
e	Eraser:	Radiergummi in zwei Größen
f	Fill:	Füll-Funktion um umschlossene Grafikausschnitte mit einem Muster zu füllen
k	Kreis:	Kreise werden gezeichnet. Mit der Taste »p« ist der Mittelpunkt des Kreises zu setzen. Danach bewegt man dann die Maus nach rechts oder links (nicht nach oben oder unten), um den Radius zu bestimmen. Drücken der linken Maustaste bedeutet dem Programm, den Kreis zu zeichnen.
l	Linie:	Linien können gezogen werden. Die Linie wird immer vom zuletzt gesetzten Punkt aus gezogen. Die Taste dient dazu, den Stift abzusetzen, um einen neuen Startpunkt anzuwählen.
p	Pencil:	Stift zum Freihandzeichnen
s	Square:	Zeichnet Quadrate oder Rechtecke. Wieder wird mit der Taste »p« der linke obere Eckpunkt gesetzt und mit einem Mausklick der linken Maustaste der rechte untere Eckpunkt angewählt.
3	Dir:	Das Inhaltsverzeichnis der Diskette mit den Bildern (*.pic) wird angezeigt und zur Eingabe eines Dateinamens aufgefordert. Die Erweiterung *.pic wird nicht mit eingegeben.
4	Save:	Hier kann das Bild gespeichert werden. Der acht Buchstaben große Dateiname erhält automatisch die Erweiterung *.pic.
5	Muster:	Hervorholen der Mustergalerie. Das ausgewählte Muster wird fortan zum Flächenfüllen verwendet.
6	Hdcopy:	Erzeugen einer Hardcopy des gezeichneten Bildes (entsprechend [Shift-PrtSc]). Vorher muß das DOS-Programm »graphics.com« geladen sein.
1	Swap1:	Ein- und Ausblenden der ersten Bildschirmseite.
2	Swap2:	Desgleichen mit der zweiten Bildschirmseite.
7	Ende:	Programmende

Tabelle. Die Kommandos des Programms »mauspain. bas«

```
[3A7] 1 DEF SEG=0:DEFINT A-Z:REM Mouse painting V8.0
[54C] 2 MSEG=256*PEEK(51*4+3)+PEEK(51*4+2) 'Testroutine
      ,ob Maustreiber geladen wurde
[545] 3 MOUSE=256*PEEK(51*4+1)+PEEK(51*4+2)
[VA4] 4 IF MSEG OR (MOUSE-2) THEN 6
[5Y5] 5 PRINT "Mouse Driver not found":END
[962] 6 DEF SEG=MSEG
[RX9] 7 IF PEEK (MOUSE-2)=207, THEN 5:'Mouse driver is
      there,continue
[4D4] 8 '***** Vorspann *****
[WMD] 9 ON ERROR GOTO 10000:CLS:SCREEN 2:KEY OFF:LOCATE
      17,26:PRINT "      Advanced Version 8.0 "
[EES] 10 LOCATE 15,26:PRINT " (c) by U.Bergmann 1987":
      LOCATE 12,26:CIRCLE (320,100),160:LINE (0,0)-(639
      ,199),1,B:LINE (180,80)-(440,100),1,B:LINE (200,1
      23)-(420,123),1:CIRCLE (320,100),235
[UU3] 20 FOR T=225 TO 185 STEP -2
[4D0] 23 '
[AA3] 25 IF A$="." THEN GOTO 30
[3A1] 26 READ A$
[902] 27 PRINT A$+" ";
[LM2] 30 CIRCLE (320,100),T
[851] 40 NEXT
[CN5] 50 DATA M,o,u,s,e," ",p,a,i,n,t,i,n,g,.
[N83] 60 FOR T=1 TO 5000:NEXT
[HRA] 150 '**** Dimensionierung der Variablen f. get/p
      ut-Befehle *****
```

Listing. »mauspain. bas«: ein superschnelles Malprogramm in Basic


```
[9A2] 160 'Dim a,b,c,...
[ZL5] 200 DIM RUBBERS(200):DIM BRUSHS(200)
[UF5] 180 DIM RUBBERB(200):DIM BRUSHB(200)
[5M5] 190 DIM PIC(8000):DIM PIC2(8000):MUS="1"
[KY5] 200 *** Draw and get the small tools ***
[3L6] 210 'LINE -(1,1)-(10,7),1,B:PAINT (3,3),0:'rubber
b

[FK4] 220 GET (0,0)-(28,12),RUBBERB:CLS
[KN5] 230 CIRCLE (5,5),5:PAINT (5,5),1:'brushs
[WA4] 240 GET (1,1)-(9,9),BRUSHS:CLS
[FM5] 250 *** Draw and get the big tools ***
[TW4] 260 'LINE (1,1)-(43,20),1,B:'rubbers
[SM4] 270 GET (0,0)-(9,5),RUBBERS:CLS
[B96] 280 CIRCLE (10,10),10:PAINT (10,10),1:'brushb
[TN5] 290 GET (1,1)-(19,19),BRUSHB:CLS:ABF$="w"
[QD3] 300 *****
[305] 310 'Draw and define work sheet
[AH0] 320 '
[K19] 330 LOCATE 3,1:PRINT " Brush =B":LOCATE 4,1:PRINT
" Clear =C"
[K19] 340 LOCATE 5,1:PRINT " Eraser=E":LOCATE 6,1:PRINT
" Fill =F"
[UZ8] 350 LOCATE 7,1:PRINT " Kreis =K":LOCATE 8,1:PRINT
" Line =L"
[UH9] 360 LOCATE 9,1:PRINT " Pencil=P":LOCATE 10,1:PRIN
T " Square=S"
[VN8] 370 LOCATE 11,1:PRINT "-----":LOCATE 12,1:PR
INT " Load =3"
[SE9] 380 LOCATE 13,1:PRINT " Save =4":LOCATE 14,1:PRI
NT " Muster=5"
[FC9] 390 LOCATE 15,1:PRINT " Print =6":LOCATE 16,1:PRI
NT " Quit =7"
[XG8] 400 LOCATE 17,1:PRINT "-----":LOCATE 20,1:PR
INT "-----"
[ZM4] 410 LOCATE 18,1:PRINT " small= 0"
[BG4] 420 LOCATE 19,1:PRINT " big = 1"
[QCB] 430 LOCATE 23,12:PRINT "Tool in use: Sta
tus: ":LOCATE 23,41:PRINT " "
[PD5] 435 LOCATE 21,1:PRINT " Muster ";ABF$
[A44] 436 LOCATE 22,1:PRINT " ";
[G44] 437 LOCATE 23,1:PRINT " ";
[7Z7] 440 LOCATE 23,48:PRINT "Horizontal: Vertika
l: "
[NH9] 442 IF ABF$="b" THEN LINE (18,170)-(66,183),1,B:P
AINT (20,180),0:GOTO 450
[Z9A] 443 IF ABF$<>"w" THEN LINE (18,170)-(66,183),1,B:
PAINT (20,180),MUS:GOTO 450
[614] 445 LINE (18,170)-(66,183),1,BF
[KS4] 450 LINE (88,9)-(630,168),1,B: Sheet
[E38] 460 LINE (0,9)-(80,190),1,B:LINE (0,0)-(639,192),
1,B: Toolbox
[XP3] 465 IF XX=1 THEN RETURN
[XP3] 470 *****
[VGA] 480 'Set max. and min. Horizontal & Vertikal Posi
tion:Show cursor
[GE6] 490 M1=4:M3=150:M4=50:CALL MOUSE(M1,M2,M3,M4)
[V94] 500 'M3=90:M4=627:'Horizontal
[WH6] 510 M1=7:M3=90:M4=627:CALL MOUSE (M1,M2,M3,M4)
[5Y3] 520 M3=10:M4=167:'Vertikal
[LS4] 530 M1=8:CALL MOUSE (M1,M2,M3,M4)
[T27] 540 M1=1:CALL MOUSE (M1,M2,M3,M4):'Show cursor
[ASS] 550 *** Hauptmenue *****
[ELC] 555 'M1=2:CALL MOUSE (M1,M2,M3,M4):GET (89,10)-(6
29,167),PIC:M1=1:CALL MOUSE(M1,M2,M3,M4)
[146] 560 'Tool choose and cursor position
[5CD] 570 AS=INKEY$:TOOL$=" ":STATUS=0:LOCATE 23,2
5:PRINT TOOL$:LOCATE 23,41:PRINT STATUS
[4W3] 580 IF AS="7" THEN GOTO 1730
[BC4] 582 IF AS="1" THEN PUT (88,9),PIC
[AH4] 584 IF AS="2" THEN PUT (88,9),PIC2
[FL5] 590 IF AS="b" THEN GOSUB 7000:GOSUB 750
[UGA] 600 IF AS="c" THEN GOSUB 7000:M1=2:CALL MOUSE(M1,
M2,M3,M4):CLS:GOTO 300
[8L5] 610 IF AS="p" THEN GOSUB 7000:GOSUB 940
[FM5] 620 IF AS="l" THEN GOSUB 7000:GOSUB 1050
[PM5] 630 IF AS="e" THEN GOSUB 7000:GOSUB 1170
[FM5] 640 IF AS="f" THEN GOSUB 7000:GOSUB 1420
[QGS] 650 IF AS="4" THEN GOSUB 7000:GOTO 1480
[XF5] 660 IF AS="3" THEN GOSUB 7000:GOTO 1560
[FG5] 665 IF AS="6" THEN GOSUB 7000:GOTO 1640
[XE5] 670 IF AS="5" THEN GOSUB 7000:GOTO 2000
[BN5] 675 IF AS="s" THEN GOSUB 7000:GOSUB 4000
[HN5] 678 IF AS="k" THEN GOSUB 7000:GOSUB 5000
[Q63] 680 GOSUB 690:GOTO 540
[V43] 690 *****
[5E3] 700 'Cursor position
[FH0] 710 '
[FS4] 720 M1=3:CALL MOUSE (M1,M2,M3,M4)
[V87] 730 LOCATE 23,59:PRINT M3:LOCATE 23,75:PRINT M4;
[GM1] 740 RETURN
[UZ8] 750 ST$="0":TOOL$="Brush":GOSUB 1370:***** Brus
h *****
[PR3] 760 'Set brush to screen
```

```
[D32] 770 ST$=INKEY$
[C34] 780 IF ST$="1" THEN STATUS=1
[C34] 790 IF ST$="0" THEN STATUS=0
[BS4] 800 M1=3:CALL MOUSE (M1,M2,M3,M4)
[L5A] 810 IF M2 AND 1 THEN M1=2:CALL MOUSE(M1,M2,M3,M4)
:GOSUB 1370:GOSUB 840
[N24] 820 IF M2 AND 2 THEN RETURN
[V53] 830 GOSUB 690:GOTO 760
[7K3] 840 IF STATUS=0 THEN 865
[PE6] 850 M1=7:M3=90:M4=611:CALL MOUSE(M1,M2,M3,M4)
[XE6] 860 M1=8:M3=5:M4=155:CALL MOUSE(M1,M2,M3,M4)
[CS1] 863 GOTO 870
[3NC] 865 M1=7:M3=90:M4=620:CALL MOUSE(M1,M2,M3,M4):M1=
8:M3=8:M4=161:CALL MOUSE(M1,M2,M3,M4)
[6V1] 870 GOSUB 690
[709] 880 IF STATUS=0 THEN IF M2 AND 1 THEN PUT (M3,M4)
,BRUSHS,OR
[XZ8] 890 IF STATUS=1 THEN IF M2 AND 1 THEN PUT (M3,M4)
,BRUSHB,OR
[B63] 900 IF M2 AND 1 THEN 870
[4K6] 910 M1=7:M3=90:M4=627:CALL MOUSE (M1,M2,M3,M4)
[8H6] 920 M1=8:M3=10:M4=167:CALL MOUSE (M1,M2,M3,M4)
[Z36] 930 M1=1:CALL MOUSE(M1,M2,M3,M4):RETURN
[NA8] 940 TOOL$="Pencil":GOSUB 1370:***** Pencil *****
[WX3] 950 'Set pencil to screen
[2K0] 960 '
[ERA] 970 IF M2 AND 1 THEN M1=2:CALL MOUSE(M1,M2,M3,M4)
:PSET (M3,M4),1:GOSUB 1000
[934] 980 IF M2 AND 2 THEN RETURN
[F63] 990 GOSUB 690:GOTO 950
[UP4] 1000 M1=3:CALL MOUSE(M1,M2,M3,M4)
[7X3] 1010 LINE -(M3,M4):GOSUB 690
[8E3] 1020 IF M2 AND 1 THEN 1000
[2R4] 1030 M1=1:CALL MOUSE(M1,M2,M3,M4)
[8V1] 1040 RETURN
[7R9] 1050 PSET (89,10),0:TOOL$="Line":GOSUB 1370:*****
Line *****
[FN0] 1060 '
[EX8] 1070 IF M2 AND 1 THEN M1=2:CALL MOUSE(M1,M2,M3,M4)
:GOSUB 1120
[E92] 1080 AB$=INKEY$
[EY4] 1090 IF AB$="p" THEN PSET (M3,M4),1
[A54] 1100 IF M2 AND 2 THEN RETURN
[ME3] 1110 GOSUB 690:GOTO 1060
[S44] 1120 M1=3:CALL MOUSE(M1,M2,M3,M4)
[KX3] 1130 LINE -(M3,M4):GOSUB 690
[SE3] 1140 IF M2 AND 1 THEN 1120
[DR4] 1150 M1=1:CALL MOUSE(M1,M2,M3,M4)
[LV1] 1160 RETURN
[QF8] 1170 TOOL$="Eraser":GOSUB 1370:***** Eraser ***
*****
[6Z5] 1180 ST$=INKEY$:Set eraser to screen
[8C4] 1190 IF ST$="1" THEN STATUS=1
[2B4] 1200 IF ST$="0" THEN STATUS=0
[Y66] 1210 IF M2 AND 1 THEN GOSUB 1370:GOSUB 1240
[N54] 1220 IF M2 AND 2 THEN RETURN
[AF3] 1230 GOSUB 690:GOTO 1180
[AX3] 1240 IF STATUS=0 THEN 1265
[DK8] 1245 M1=2:CALL MOUSE(M1,M2,M3,M4):REM großer Radi
erer
[KR6] 1250 M1=7:M3=88:M4=601:CALL MOUSE(M1,M2,M3,M4)
[LF6] 1260 M1=8:M3=9:M4=155:CALL MOUSE(M1,M2,M3,M4)
[SW1] 1261 GOTO 1270
[Q88] 1265 M1=2:CALL MOUSE(M1,M2,M3,M4):REM kleiner Rad
ierer
[1S6] 1266 M1=7:M3=89:M4=620:CALL MOUSE(M1,M2,M3,M4)
[MP6] 1267 M1=8:M3=10:M4=162:CALL MOUSE(M1,M2,M3,M4)
[GY1] 1270 GOSUB 690
[XN0] 1280 '
[L72] 1290 'GOSUB 1060
[ZW6] 1300 IF STATUS=0 THEN PUT (M3,M4),RUBBERS,AND
[2W6] 1310 IF STATUS=1 THEN PUT (M3,M4),RUBBERB,AND
[1F3] 1320 IF M2 AND 1 THEN 1270
[BR4] 1330 M1=1:CALL MOUSE(M1,M2,M3,M4)
[AR6] 1340 M1=7:M3=90:M4=627:CALL MOUSE(M1,M2,M3,M4)
[VP6] 1350 M1=8:M3=10:M4=167:CALL MOUSE(M1,M2,M3,M4)
[TV1] 1360 RETURN
[SW2] 1370 *****
[DZ3] 1380 'Tooloutput & Status
[YH4] 1390 LOCATE 23,25:PRINT TOOL$
[FU4] 1400 LOCATE 23,41:PRINT STATUS
[8V1] 1410 RETURN
[ZP2] 1420 *****
[FP2] 1430 'Fill picture
[MR4] 1440 M1=2:CALL MOUSE(M1,M2,M3,M4)
[896] 1450 M1=3:CALL MOUSE(M1,M2,M3,M4):GOTO 1452
[BUC] 1451 LOCATE 22,2:PRINT " ":LOCATE 23,2:PR
INT " ":LINE (18,170)-(66,183),1,B:RETURN
[LTA] 1452 IF ABF$="w" THEN GOSUB 1451:PAINT (20,180),1
:PAINT (M3,M4),1:RETURN
[XPA] 1454 IF ABF$="b" THEN GOSUB 1451:PAINT (20,180),0
:PAINT (M3,M4),0:RETURN
```



```

[922] 1460 PAINT (M3,M4),MU$
[ØW1] 1470 RETURN
[KT6] 1472 '***** Save Routine *****
*****
[C5A] 1480 SX$="":LOCATE 23,12:PRINT "
";
[2BC] 1482 LOCATE 23,12:INPUT "<< Graphik speichern >>
Abbruch=Enter Name: ";SX$
[A2F] 1484 IF SX$="" THEN DEF SEG=MSEG:LOCATE 23,12:PRI
NT "
";:GOTO 3000
[9S4] 1485 M1=2:CALL MOUSE(M1,M2,M3,M4)
[CA8] 1486 DEF SEG=&HB800:BSAVE SX$+".pic",Ø,16200:DEF
SEG=MSEG
[AS4] 1488 M1=1:CALL MOUSE(M1,M2,M3,M4)
[AS4] 1490 M1=3:CALL MOUSE(M1,M2,M3,M4)
[APA] 1492 LOCATE 23,12:PRINT "
";:GO
TO 3000
[UR6] 1500 '***** Load picture *****
*****
[XMG] 1500 SS$="":M1=2:CALL MOUSE(M1,M2,M3,M4):GET (88,9
)-(630,168),PIC:CLS:LOCATE 2,10:PRINT " Pictures
auf Diskette "
[LK3] 1591 LINE (70,5)-(265,19),1,B
[DT4] 1592 LOCATE 4,4:FILES "*.pic"
[9T6] 1593 LINE (Ø,Ø)-(639,190),1,B:LINE (Ø,168)-(630,1
68),1
[KD9] 1594 LOCATE 23,12:PRINT "
";
[Ø2B] 1595 LOCATE 23,12:INPUT "<< Graphik laden >> A
bbruch=Enter Name: ";S$
[H3C] 1596 IF S$="" OR S$="****" THEN DEF SEG=MSEG:LOCAT
E 23,55:CLS:PUT (88,9),PIC:GOTO 3000
[Z47] 1600 DEF SEG=&HB800:BLOAD S$+".pic":DEF SEG=MSEG
[4L6] 1602 GET (88,9)-(630,168),PIC:CLS:PUT (88,9),PIC
[CR4] 1610 M1=1:CALL MOUSE(M1,M2,M3,M4)
[MR4] 1620 M1=3:CALL MOUSE(M1,M2,M3,M4)
[PNA] 1630 LOCATE 23,12:PRINT "
";:GO
TO 3000
[687] 1640 '***** Print picture *****
*****
[289] 1642 LOCATE 23,12:PRINT "
";
[USC] 1643 LOCATE 23,12:PRINT "<< Grafik drucken >>
S=Start Abbruch=Enter ";
[M92] 1644 AB$=INKEY$
[CØ4] 1645 IF AB$=CHR$(13) THEN 3000
[XB3] 1646 IF AB$="s" THEN 1652
[CX1] 1647 GOTO 1644
[YR4] 1650 M1=2:CALL MOUSE(M1,M2,M3,M4)
[WH2] 1651 'DIM PIC(8000)
[SL6] 1652 GET (88,9)-(630,168),PIC:CLS:PUT (88,9),PIC
[NM1] 1660 H.C.R=Ø
[RE4] 1670 HARD.COPY=VARPTR(H.C.R)
[TR3] 1680 POKE HARD.COPY+Ø,205
[4G3] 1690 POKE HARD.COPY+1,5
[WP3] 1700 POKE HARD.COPY+2,203
[EZ2] 1710 CALL HARD.COPY
[426] 1720 M1=1:CALL MOUSE(M1,M2,M3,M4):GOTO 3000
[ZV6] 1730 '***** Quit routine *****
*****
[1D9] 1735 LOCATE 23,12:PRINT "
";
[PAD] 1740 LOCATE 23,12:PRINT "<< Programm beenden >>
(j=beenden / Enter=zurück) ";AB$;
[R92] 1750 AB$=INKEY$
[S53] 1755 IF AB$="" THEN 1750
[WFD] 1758 IF AB$=CHR$(13) THEN LOCATE 23,12:PRINT "
";:GOTO 3000
[5FA] 1760 IF AB$="j" THEN CLS:LOCATE 7,30:PRINT "Mouse
painting ends..."
[8PP] 1770 LOCATE 9,30:PRINT "Verbesserungskritiken am"
;:LOCATE 10,30:PRINT "Programm bitte an ";:LOCATE
12,30:PRINT "U.Bergmann";:LINE (Ø,Ø)-(639,199),1
,B
[637] 1771 LOCATE 8,30:PRINT "===== ";
[CS7] 1772 LOCATE 13,30:PRINT "Kaiserslautererstr.10";
[LR6] 1774 LOCATE 14,30:PRINT "7000 Stuttgart 31";
[F17] 1776 LOCATE 16,30:PRINT "schreiben. Danke!";
[1F7] 1778 LOCATE 18,30:PRINT ">> Have a nice day ! << "
";
[DS4] 1780 M1=2:CALL MOUSE(M1,M2,M3,M4)
[5V3] 1790 FOR T=200 TO 220 STEP 2
[FNØ] 1800 '
[283] 1810 CIRCLE (325,100),T
[ØA1] 1820 NEXT
[VNØ] 1830 '
[ZNØ] 1840 '
[E64] 1850 END 'The end...(really)

```

```

[YZ3] 2000 '***** Muster *****
[ØU4] 2100 M1=2:CALL MOUSE (M1,M2,M3,M4)
[5C4] 2200 GET (88,9)-(630,168),PIC:CLS
[GFC] 2300 LOCATE 4,20:PRINT "Musterwahl fuer Fill-F
unktion":LINE (Ø,Ø)-(639,192),1,B
[17Y] 2400 AS$="LIST " :BS$="T-D"+CHR$(128):CS$="342GE":DS$=
STRINGS(3,128)+"-":ES$=STRINGS(3,128):FS$="55++55++
55++33--33++55"+STRINGS(2,128):GS$="*****++*****
+-----":HS$=STRINGS(2,128)+CHR$(4)+"!&H6D77"+CHR$(
128):IS$="++++-+-----+---+---":JS$=AS$+CS$+BS$+GS$
[L26] 2500 LINE (100,50)-(140,70),1,B:PAINT (110,55),AS
[V26] 2600 LINE (200,50)-(240,70),1,B:PAINT (210,55),BS
[336] 2700 LINE (300,50)-(340,70),1,B:PAINT (310,55),CS
[B36] 2800 LINE (400,50)-(440,70),1,B:PAINT (410,55),DS
[446] 2850 LINE (500,50)-(540,70),1,B:PAINT (510,55),ES
[ZF6] 2852 LINE (100,90)-(140,110),1,B:PAINT (110,100),
FS
[9G6] 2854 LINE (200,90)-(240,110),1,B:PAINT (210,100),
GS
[LG6] 2856 LINE (300,90)-(340,110),1,B:PAINT (310,100),
HS
[XG6] 2858 LINE (400,90)-(440,110),1,B:PAINT (410,100),
IS
[1H6] 2860 LINE (500,90)-(540,110),1,B:PAINT (510,100),
JS
[MM6] 2870 LINE (100,130)-(140,150),1,B:PAINT (110,140),
1
[VM6] 2880 LINE (200,130)-(240,150),1,B:PAINT (210,140),
Ø
[FL3] 2900 LOCATE 8,8:PRINT "1"
[YT3] 2910 LOCATE 8,21:PRINT "2"
[BU3] 2920 LOCATE 8,34:PRINT "3"
[TU3] 2925 LOCATE 8,47:PRINT "4"
[VU3] 2930 LOCATE 8,60:PRINT "5"
[U3] 2935 LOCATE 13,8:PRINT "6"
[Ø24] 2940 LOCATE 13,21:PRINT "7"
[C24] 2942 LOCATE 13,34:PRINT "8"
[S24] 2944 LOCATE 13,47:PRINT "9"
[634] 2946 LOCATE 13,60:PRINT "A"
[YW3] 2948 LOCATE 17,8:PRINT "W"
[H34] 2950 LOCATE 17,21:PRINT "B"
[V58] 2955 LOCATE 23,12:PRINT "
Abbruch=Enter";
[NC2] 3000 ABF$=INKEY$
[YMØ] 3100 '
[PY3] 3200 IF ABF$="1" THEN MU$=AS
[6Z3] 3220 IF ABF$="2" THEN MU$=BS
[PZ3] 3240 IF ABF$="3" THEN MU$=CS
[6Ø4] 3260 IF ABF$="4" THEN MU$=DS
[KØ4] 3270 IF ABF$="5" THEN MU$=ES
[ZØ4] 3275 IF ABF$="6" THEN MU$=FS
[814] 3276 IF ABF$="7" THEN MU$=GS
[K14] 3278 IF ABF$="8" THEN MU$=HS
[Y14] 3288 IF ABF$="9" THEN MU$=IS
[334] 3289 IF ABF$="a" THEN MU$=JS
[D44] 3290 IF ABF$="w" THEN MU$="w"
[614] 3292 IF ABF$="b" THEN MU$="b"
[TA3] 3300 IF ABF$="" THEN 3000
[TX4] 3350 IF ABF$=CHR$(13) THEN ABF$="w"
[8PA] 3400 CLS:PUT (88,9),PIC:LINE (18,170)-(66,183),1,
B:PAINT (20,180),MU$:GOTO 3000
[YS4] 3900 '*****
[LLA] 4000 X1=89:Y1=10:TOOL$="Square":GOSUB 1370:***
Box Funktion *****
[R82] 4100 AB$=INKEY$
[EØ6] 4200 IF AB$="p" THEN PSET (M3,M4),1:X1=M3:Y1=M4
[47F] 4300 IF M2 AND 1 THEN M1=2:CALL MOUSE(M1,M2,M3,M4
):LINE (X1,Y1)-(M3,M4),1,B:M1=1:CALL MOUSE (M1,M2
,M3,M4)
[ØY1] 4400 GOSUB 690
[W54] 4500 IF M2 AND 2 THEN RETURN
[EW1] 4600 GOTO 4100
[K93] 4700 '*****
[TG9] 5000 TOOL$="Kreis":GOSUB 1370:***** Kreis ***
*****
[A92] 5050 AB$=INKEY$
[7Z8] 5100 IF AB$="p" THEN PSET (M3,M4),1:PSET (X1,Y1),
Ø:X1=M3:Y1=M4
[6D2] 5150 R=ABS(M3-X1)
[U1V] 5200 IF M2 AND 1 THEN X2=ABS(M3):M1=2:CALL MOUSE(
M1,M2,M3,M4):CIRCLE (X1,Y1),R:PSET (X1,Y1),Ø:GOSU
B 6000:XX=1:GOSUB 3000:TOOL$="Kreis":GOSUB 1370:M1
=1:CALL MOUSE (M1,M2,M3,M4):YY=Ø
[ZX1] 5300 GOSUB 690
[NX4] 5400 IF M2 AND 2 THEN XX=Ø:RETURN
[ENØ] 5500 '
[YW1] 5510 GOTO 5050
[4Ø8] 6000 GET (88,9)-(630,168),PIC:CLS:PUT (88,9),PIC:
RETURN
[B3E] 7000 M1=2:CALL MOUSE (M1,M2,M3,M4):GET (89,10)-(6
29,167),PIC2:M1=1:CALL MOUSE(M1,M2,M3,M4):RETURN
[VK5] 9000 '***** Error Routine *****
[TS5] 10000 IF ERR=53 THEN S$="****":RESUME 1593
[U82] 10100 RESUME 3000

```

Listing. »mauspain. bas«: ein superschnelles Malprogramm in Basic

Eine große Sache – Minimouse

Das in Turbo Pascal 3.0 geschriebene, speicherresidente Programm »Minimouse« ist der richtige Speck für Ihre Maus. Von nun an können Sie fast jedes Anwendungsprogramm mit der Maus steuern. Damit ist Ihre Maus nicht länger für die Katz.

Das Programm »Minimouse« ist ein Zusatzmaustreiber für Mouse-Systems-Mode-kompatible Mäuse (Genius GM6, Witty Mouse, Summamouse oder ähnliches). Minimouse funktioniert nur, wenn vorher der entsprechende Mouse-Systems-Mode-Treiber geladen wird. MS-Mouse-kompatible Mäuse werden nicht unterstützt. Durch die Kombination von Maustreiber und Bildschirmditor lassen sich fast alle Anwendungsprogramme, wie zum Beispiel Turbo Pascal 3.0 und 4.0, GW-Basic, Editoren, Textverarbeitungsprogramme und Lotus 1-2-3, mit der Maus steuern. Bei einigen Spielen, bei denen der Timer-Interrupt nicht weitergeleitet wird, funktioniert das Programm leider nicht, da es sich hierbei um einen timergesteuerten TSR-Prozeß handelt (Terminate but Stay Resident: Ein Programm, das nach der Aktivierung im Speicher verbleibt). Dieser läuft gleichzeitig mit dem Anwendungsprogramm im Hintergrund ab. Grundlagen hierzu sind dem Kurs »Speicherresidente Programme selbstgemacht« (DOS International, Ausgabe 2'88 bis 5'88) zu entnehmen. Das Programm ist für die Farbgrafikkarte (CGA) geschrieben, läßt sich aber durch Ändern der Variablen »video« in B000hex statt B800hex an die Hercules-Karte anpassen.

Der Bildschirmditor:

Nach dem Start befindet sich das Programm im Bildschirmditor-Modus. Dieser läßt sich auch vom Maus-Modus aus durch Drücken der rechten Maustaste aktivieren. Auf dem Bildschirm erscheint beim Bewegen der Maus ein Cursor, der den Mausbewegungen folgt. Um Zeilen vom Bildschirm in den Tastaturpuffer zu kopieren, drückt man die linke Maustaste und fährt gleichzeitig mit der Maus nach rechts. Die Zeichen, die vom Cursor überstrichen werden, gelangen nun in den Tastaturpuffer. Sie stehen somit dem laufenden Anwendungsprogramm oder dem Betriebssystem zur Verfügung. Der Bildschirmditor funktioniert nur im Textmodus.

Der Maus-Modus:

Das Drücken der mittleren Maustaste aktiviert den Maus-Modus. Die Bewegungen der Maus werden hierbei in mehrere Tastaturbetätigungen umgewandelt. Wird die Maus schneller bewegt, so vergrößert sich die Zahl der Tastaturbetätigungen (tatsächliche Mausfunktion; kein Joystickeffekt). Per Voreinstellung sind hierfür die Cursor-Tasten vorgesehen. Die linke Maustaste ist die Aktionstaste und entspricht der Return-Taste.

Die Umsetzung der Mausbewegungen in Tastaturbetätigungen geschieht durch direktes Schreiben in den Tastaturpuffer. Hierbei ist die Tastenposition der zu simulierenden Taste, die als zweites Byte eines Wortes im Tasta-

Modus	links	mitte	rechts	Aktion
Maus	X			Aktionstaste
Maus	X		X	Programm AN
Maus		X	X	Programm AUS
Maus			X	Bildschirm-Editor-Modus
Maus	X	X		Spiel-Modus AN
Maus		X		Spiel-Modus AUS
Editor		X		Maus-Modus
Editor	X			Zeichen kopieren

X = gedrückte Maustaste

Die Tabelle faßt die einzelnen Maustasten-Funktionen zusammen

turpuffer stehen muß (siehe DOS International, Ausgabe 4'88, Seite 144), genauso wichtig wie der ASCII-Wert der Taste, der als erstes Byte des Wortes im Puffer steht. Dies ist deshalb von Bedeutung, da einige Programme die Tastenposition zusätzlich, andere dagegen die Position ausschließlich abfragen. Bei allen Steuertasten (Cursor-Tasten, [PgUp] und [PgDn]) ist der ASCII-Wert gleich Null zu setzen.

Der Spiel-Modus:

Bei Spielen muß der Spiel-Modus durch gleichzeitiges Drücken der linken und mittleren Maustaste aktiviert werden. Spiele lassen sich nun viel genauer steuern, da – unabhängig davon, wie schnell man die Maus bewegt – immer nur eine Tastenbetätigung erzeugt wird (Joystickeffekt). Die Aktionstaste ist in diesem Modus nicht entprellt. Sie läßt sich somit als »Dauerfeuertaste« benutzen. Man verläßt den Spiel-Modus, indem man die mittlere Maustaste betätigt.

Aufruf und Initialisierung:

Das Programm ruft man entweder mit »mini [Return]« auf, um mit der Voreinstellung zu arbeiten, oder mit »mini i [Return]«, wobei die Initialisierungsroutine gestartet wird. Hier bestimmt der Benutzer die Tasten, die später im Maus-Modus durch die Maus angesprochen werden. Es lassen sich also auch Programme steuern, die nicht per Cursor-Tasten bedient werden. Die Aktionstaste ist bei der Initialisierung ebenfalls festzulegen, beispielsweise bei Spielen als Feuertaste.

Aber Achtung, vor jedem erneuten Aufruf des Programms sollte der Computer durch einen Warmstart (Reset) zurückgesetzt werden, denn wenn das Programm Minimouse mehrfach mit verschiedenen Initialisierungen im Speicher steht, kann dies zu Komplikationen führen. Der Rechner »stürzt« dabei jedoch nicht ab.

Abstellen des Programms:

Ist das Programm Minimouse aktiviert, so ergeben sich Probleme mit Programmen, die die Maus bereits selbst unterstützen. Um diese Schwierigkeiten zu unterbinden, läßt sich das Programm Minimouse – sofern es sich im Maus-Modus befindet – durch gleichzeitiges Drücken der linken und rechten Maustaste quasi »auf Eis legen«. Drückt man die mittlere und die rechte Maustaste, so ist das Programm wieder aktiv.

Ein Stück (Programmierer-) Lebensfreude – Turbo Pascal 4.0 mit Mausbedienung

Zuerst muß das Programm Minimouse aktiviert werden. Anschließend starten Sie Turbo Pascal 4.0.

Die Windows im Maus-Modus:

Es ist kein Cursor auf dem Bildschirm zu sehen. Durch Verschieben der Maus nach links oder rechts werden die Windows geöffnet. Die Auswahl der gewünschten Aktion erfolgt durch Auf- oder Abbewegen der Maus. Will man eine gewählte Aktion auslösen, drückt man die Aktionstaste.

Die Windows im Bildschirmditor-Modus:

Hier ist der Cursor auf den Anfangsbuchstaben des Window-Namens zu bewegen. Das Window wird durch Drücken der linken Maustaste geöffnet. Um eine Aktion auszuwählen, »klickt« man ihren Anfangsbuchstaben an.

Der Turbo-Pascal-Editor:

Will man ebenfalls im Turbo-Pascal-Editor in den Genuß einer Mausunterstützung gelangen, muß der Maus-Modus aktiv sein. Der Textcursor läßt sich dann mit der Maus be-

liebig hin- und herbewegen. Natürlich können Sie auch hier Zeichen oder Zeilen mit dem Bildschirmeditor kopieren.

Die Tabelle (Bild) gibt einen Überblick über die Maustasten-Funktionen.

(Heiko Gerlach/zi)

```
[L93] 1: PROGRAM minimouse;
[000] 2:
[LU5] 3: CONST Video = $B800; { für CGA }
[GU3] 4: { $B000 für Hercules }
[XH3] 5: rechts : INTEGER = 77;
[PD3] 6: reascii : INTEGER = 0;
[BE3] 7: links : INTEGER = 75;
[PD3] 8: liascii : INTEGER = 0;
[R93] 9: hoch : INTEGER = 72;
[OE3] 10: hoascii : INTEGER = 0;
[HL3] 11: runter : INTEGER = 80;
[6F3] 12: ruascii : INTEGER = 0;
[KK3] 13: aktion : INTEGER = 28;
[DR3] 14: aktascii : INTEGER = 13;
[P53] 15: xpos : INTEGER = 0;
[S53] 16: ypos : INTEGER = 0;
[143] 17: stat : INTEGER = 0;
[XU2] 18: scroon : CHAR = ' ';
[TS2] 19: i : INTEGER = 0;
[Y93] 20: c : STRING[20] = '';
[MB3] 21: c2 : STRING[20] = '';
[KT3] 22: akt : BOOLEAN = false;
[9T3] 23: game : BOOLEAN = false;
[064] 24: zeichen : BOOLEAN = false;
[DX3] 25: prell : BOOLEAN = false;
[753] 26: off : BOOLEAN = false;
[583] 27: cdum : STRING[20] =
[F21] 28:
[000] 29:
[000] 30:
[FR3] 31: TYPE str = STRING[20];
[S13] 32: registers = RECORD
[283] 33: ax,bx,cx,dx,bp,si,di,
[9C3] 34: ds,es,Flags : INTEGER
[ZN0] 35: END;
[000] 36:
[FK6] 37: VAR vektoroff : INTEGER absolute $0:$70;
[HW5] 38: vektorseg : INTEGER absolute $0:$72;
[183] 39: regs : registers;
[DT2] 40: dum : CHAR;
[Z14] 41: Mouse : registers;
[9K3] 42: m1, m2, m3, m4 : INTEGER;
[7C3] 43: mouse : BOOLEAN;
[000] 44:
[KA7] 45: PROCEDURE keybuff(VAR keys; VAR posi:str);
[8G2] 46: CONST segm=$40;
[0A1] 47: offs=$1E;
[8E6] 48: VAR anfang : INTEGER absolute segm:$1A;
[AH5] 49: ende : INTEGER absolute segm:$1C;
[AA5] 50: s : STRING[20] absolute keys;
[MU0] 51: BEGIN
[8L5] 52: IF length(s)>15 THEN s:=copy(s,1,15);
[UL3] 53: IF length(posi)>15 THEN
[CG3] 54: posi:=copy(posi,1,15);
[V14] 55: anfang:=offs; ende:=offs;
[544] 56: WHILE length(s)>0 DO BEGIN
[NL4] 57: mem[segm:ende+0]:=ord(s[1]);
[8A5] 58: mem[segm:ende+1]:=ord(posi[1]);
[102] 59: ende:=ende+2;
[SU3] 60: s:=copy(s,2,length(s)-1);
[3D5] 61: posi:=copy(posi,2,length(posi)-1)
[4N0] 62: END;
[ZN0] 63: END;
[000] 64:
[000] 65:
[X57] 66: procedure install(text:str; VAR richtung,
[5L2] 67: ascii:INTEGER);
[8G2] 68: CONST segm=$40;
[0A1] 69: offs=$1E;
[8E6] 70: VAR anfang : INTEGER absolute segm:$1A;
[AH5] 71: ende : INTEGER absolute segm:$1C;
[RS1] 72: ch : CHAR;
[MU0] 73: BEGIN
[HH6] 74: writeln('Taste für ',text,' drücken');
[S02] 75: read(kbd,ch);
[XC2] 76: ascii:=ord(ch);
[V14] 77: anfang:=offs; ende:=offs;
```

Das gab's noch nie!

Das Beste aus PASCAL 11/86 bis 6/87 im:

PASCAL SONDERDRUCK



Über 200 Seiten Pascal-Programme für Hobby, Studium und Beruf, Lernen und Unterricht in einmaliger Mischung und Zusammenstellung in einem einzigen Heft:

Eine Wanderung durch die Landschaft der Programmiersprachen · Kurvendiskussion mit grafischer Darstellung · Mathematische Funktionen selbst erweitert · Rechnen mit komplexen Zahlen in Turbo Pascal · Datenstrukturen mit Zeigern, Darstellung und Beispiel · Gleichungssysteme und Ausgleichspolynome · Strings unter Wirth'schem Standard selbst gemacht · Press & Unpress, Dateien komprimiert abspeichern · GSX – Die unbekannte Größe bei CP/M 3.0 · Source-Code Referenz Lister · Dreidimensionale Darstellungen in Pascal ST+ (Atari ST) · Bibliothek zum Rechnen mit Matrizen in Turbo Pascal · Simulation in Pascal, ein Fahrkartenautomat · Unison – Ähnlichkeitsbestimmung von Zeichenketten · Sichere Eingabe von Real- und Integerzahlen · Komfortable Stringeingabe mit Editiermöglichkeit · Pascal Colleg – Schleifen, Operatoren, logische Werte · Vom Punkt zur dritten Dimension, Grafik selbst gemacht · Zweidimensionale Meßwertdarstellung in Pascal · Plotter-Simulator für den Matrixdrucker · Externe Kommandos in Turbo Pascal 3.0 · Setup – Menuegesteuerte Druckereinstellung · Datenfernübertragungs-Programm in Pascal · Standardisierter Menue-Aufbau · Richtiges Runden in Turbo Pascal · HGR-Copy: Bildschirmausschnitte und Zoom bei Hardcopies · Interrupts – unter Turbo Pascal programmiert · Der Case-Trick, PEEK und POKE für Pascal-Programmierer

Alle Programme sind in der Praxis getestet, Nachträge und Verbesserungen wurden eingearbeitet. Sie sind, mit einer Ausnahme, unter Turbo Pascal 3.0 lauffähig und eignen sich für alle Hardware und Betriebssysteme vom Homecomputer bis zum Mini.

PASCAL Sonderdruck, Bestellung beim DMV-Verlag
Best.-Nr. 345

Inland:		Ausland:	
Einzelpreis	25,- DM	Einzelpreis	25,- DM
zzgl. Versandkosten	3,- DM	zzgl. Versandkosten	5,- DM
Endpreis	28,- DM	Endpreis	30,- DM

Alle Programme sind auf Datenträger in der abgedruckten Version erhältlich:

PASCAL Sonderdruck Volume 1 und 2

Best.-Nr. 346 MS-DOS – 5 1/4" Best.-Nr. 348 Atari ST – 3 1/2"
Best.-Nr. 347 MS-DOS – 3 1/2" Sonderformate: Best.-Nr. 349

Inland:		Ausland:	
Einzelpreis	35,- DM	Einzelpreis	35,- DM
zzgl. Versandkosten	3,- DM	zzgl. Versandkosten	5,- DM
Endpreis	38,- DM	Endpreis	40,- DM

— Bitte benutzen Sie die Bestellkarte —

ERHÄLTICH NUR BEI:

DMV Verlag · Postfach 250 · 3440 Eschwege


```

[7W4] 78: richtung:=MEM[segm:ende+1];
[N63] 79: if (richtung>=69) and
[9F4] 80: (richtung<=83) then ascii:=0
[ZN0] 81: end;
[000] 82:
[000] 83:
[H96] 84: PROCEDURE Get_mouse_status(Var m2, m3,
[Z22] 85: m4 : INTEGER);
[MU0] 86: BEGIN
[1D2] 87: Mouser.ax := 3;
[9W2] 88: intr ($33, mouser);
[DG2] 89: m2 := mouser.bx;
[LG2] 90: m3 := mouser.cx;
[YE2] 91: m4 := mouser.dx
[ZN0] 92: END;
[000] 93:
[2C6] 94: PROCEDURE Mouse_motion(Var xcounter,
[L83] 95: ycounter : INTEGER);
[MU0] 96: BEGIN
[ED2] 97: mouser.ax := 11;
[9W2] 98: intr ($33, mouser);
[MS3] 99: xcounter := mouser.cx;
[6L3] 100: ycounter := mouser.dx
[ZN0] 101: END;
[000] 102:
[000] 103:
[ME3] 104: PROCEDURE miniproc;
[8G2] 105: CONST segm=$40;
[0A1] 106: offs=$1E;
[KA5] 107: VAR screen : ARRAY [0..24,0..79,0..1]
[C54] 108: OF CHAR absolute video;0;
[HR5] 109: anfang : INTEGER absolute segm:$1A;
[AH5] 110: ende : INTEGER absolute segm:$1C;
[4G5] 111: dummy1,dummy2,taste : INTEGER;
[W36] 112: step1,step2,move_x,move_y: INTEGER;
[MU0] 113: BEGIN
[U33] 114: inline($50/$53/$51/$52/
[YM2] 115: $57/$56/$06/$1E/$FB);
[EU0] 116: c:='';
[D95] 117: mouse_motion(move_x,move_y);
[RT6] 118: get_mouse_status(taste,dummy1,dummy2);
[4R3] 119: IF (taste=6) THEN BEGIN
[ZX1] 120: off:=false;
[TA1] 121: xpos:=1;
[091] 122: ypos:=1
[ZN0] 123: END;
[WC3] 124: IF NOT(off) THEN BEGIN
[3M5] 125: IF ((taste=4) AND (stat=0)) THEN BEGIN
[VT1] 126: akt:=true;
[E12] 127: game:=false;
[H65] 128: screen[ypos,xpos,1]:=screon;
[4M0] 129: END
[Y83] 130: ELSE IF (taste=2) AND
[D53] 131: (stat=0) THEN BEGIN
[YR1] 132: akt:=false
[ZN0] 133: END;
[TS5] 134: IF (akt) THEN BEGIN (*Mousetreiber*)
[8F4] 135: IF taste=5 THEN game:=true;
[Y23] 136: IF (move_x>3) THEN
[MY4] 137: FOR i:=1 TO (move_x div 10) DO
[MU0] 138: BEGIN
[PV2] 139: c:=c+CHR(reascii);
[CN2] 140: c2:=c2+CHR(rechts)
[ZN0] 141: END;
[953] 142: IF (move_x<-3) THEN
[AE5] 143: FOR i:=1 TO ((move_x*-1) div 10) DO
[MU0] 144: BEGIN
[MV2] 145: c:=c+CHR(liascii);
[BG2] 146: c2:=c2+CHR(links)
[ZN0] 147: END;
[D53] 148: IF (move_y<-3) THEN
[BE5] 149: FOR i:=1 TO ((move_y*-1) div 10) DO
[MU0] 150: BEGIN
[0W2] 151: c:=c+CHR(hoascii);
[AB2] 152: c2:=c2+CHR(hoch)
[ZN0] 153: END;
[233] 154: IF (move_y>3) THEN
[BV5] 155: FOR i:=1 TO (move_y div 10) DO BEGIN
[PX2] 156: c:=c+CHR(ruascii);
[CS2] 157: c2:=c2+CHR(runter)
[ZN0] 158: END;
[FA4] 159: IF ((taste<>1) OR (game)) THEN
[372] 160: prell:=false;
[N75] 161: IF ((taste=1) AND NOT(prell)) THEN
[MU0] 162: BEGIN
[P32] 163: prell:=true;
[S23] 164: c:=c+CHR(aktascii);
[FP2] 165: c2:=c2+CHR(aktion)
[ZN0] 166: END;

```

```

[0F4] 167: IF (taste=3) then off:=true;
[U34] 168: IF c <> '' THEN keybuff(c,c2);
[CM1] 169: c:=''; c2:=''
[A93] 170: END (*Mousetreiber*)
[DE4] 171: ELSE BEGIN (*Screeneditor*)
[2F5] 172: IF (stat=0) OR (abs(move_x)>2) THEN
[HD5] 173: IF (zeichen) THEN zeichen:=false;
[7P3] 174: IF ((abs(move_x)>2) OR
[084] 175: (abs(move_y)>2) THEN BEGIN
[GA2] 176: IF (stat=0) AND
[CK3] 177: (abs(move_x)>=10) THEN
[0G3] 178: step1:=(move_x div 10)
[XA5] 179: ELSE IF (move_x<0) THEN step1:=-1
[S92] 180: ELSE step1:=1;
[3F3] 181: IF game THEN step1:=1;
[GA2] 182: IF (stat=0) AND
[CK3] 183: (abs(move_y)>=10) THEN
[5G3] 184: step2:=(move_y div 10)
[X65] 185: ELSE IF move_y<0 THEN step2:=-1
[V92] 186: ELSE step2:=1;
[6F3] 187: IF game THEN step2:=1;
[H65] 188: screen[ypos,xpos,1]:=screon;
[0T2] 189: IF ((move_x>1) AND
[PB3] 190: (xpos<80-step1)) THEN
[Y22] 191: xpos:=xpos+step1;
[523] 192: IF ((move_x<-1) AND
[FC4] 193: (xpos>=step1*-1)) THEN BEGIN
[TD2] 194: IF (stat=1) THEN
[836] 195: screen[ypos,xpos-step1,1]:=screon
;
[8W2] 196: xpos:=xpos+step1
[ZN0] 197: END;
[E43] 198: IF stat=0 THEN BEGIN
[923] 199: IF ((move_y<-1) AND
[0G3] 200: (ypos>=step2*-1)) THEN
[503] 201: ypos:=ypos+step2;
[4T2] 202: IF ((move_y>1) AND
[9B3] 203: (ypos<25-step2)) THEN
[FW2] 204: ypos:=ypos+step2
[ZN0] 205: END;
[GE6] 206: IF (screen[ypos,xpos,1]<>CHR($79)) TH
EN
[P55] 207: screon:=screen[ypos,xpos,1];
[UY4] 208: screen[ypos,xpos,1]:=CHR($79)
[ZN0] 209: END;
[GP3] 210: IF (taste=1) THEN BEGIN
[491] 211: stat:=1;
[S54] 212: c:=screen[ypos,xpos,0];
[9A4] 213: IF zeichen=true THEN c:='';
[EL4] 214: IF c<>'' THEN keybuff(c,c2);
[SA2] 215: zeichen:=true
[4M0] 216: END
[322] 217: ELSE stat:=0
[9A3] 218: END (*Screeneditor*)
[ZN0] 219: END;
[G94] 220: inline($1F/$07/$5E/$5F/$5A/$59/
[D63] 221: $5B/$58/$8B/$E5/$5D/$CF)
[ZN0] 222: END;
[000] 223:
[6K2] 224: BEGIN { main }
[LS6] 225: writeln('Co-Mouse Treiber installiert');
[826] 226: writeln('(C) 1988; DOS International '+
[7U2] 227: '& Heiko Gerlach');
[DF4] 228: IF paramstr(1)='i' THEN BEGIN
[V05] 229: install('Start',links,links);
[R95] 230: install('links',links,liascii);
[BS5] 231: install('rechts',rechts,reascii);
[7T4] 232: install('hoch',hoch,hoascii);
[ZW5] 233: install('runter',runter,ruascii);
[UW5] 234: install('Aktion',aktion,aktascii);
[NV5] 235: writeln('Initialisierung beendet')
[ZN0] 236: END;
[KU1] 237: inline($FA);
[HK4] 238: vektoroff:= ofs(miniproc);
[ZU2] 239: vektorseg:=cseg;
[NU1] 240: inline($FB);
[UA2] 241: regs.ax:=$3100;
[UA2] 242: regs.dx:=$1000;
[YX1] 243: msdos (regs)
[KN0] 244: END.

```

Listing. »mini.pas« erweitert den Einsatzbereich der Maus

Erfolgreich programmieren mit komfortablen Musterlösungen in C



inklusive Diskette

Dieses Nachschlagewerk bietet Ihnen:

- **das Know-how zum C-System: Programmierungsumgebung** (Compiler, Preprozessor, Linker, Library Manager, MAKE-Utility, Debugger...), **Sprachkonzept/-elemente** (Datentypen, Operatoren und Ausdrücke, Strukturen, Vektoren und Zeiger...), **Bibliotheksfunktionen von Microsoft- und Turbo-C** (Ein-/Ausgabe, Dateizugriff, Speicherverwaltung, Konvertierung, Stringbehandlung...);
- **Kurse zur Dateiverwaltung** Dateianzeige, Kopieren von Dateien...; **Grafik** (Zeichnen von Linien, Kreisen, Schraffieren/Invertieren sowie geometrische Transformationen...), **Software-engineering**;
- **detaillierte Betriebssystembeschreibungen** von Unix/MS-DOS: Ein-/Ausgabe (z.B. Bildschirmsteuerung), Interrupts und Signalbehandlung, Dateizugriff u.a.;
- **Utilities zur hardwarenahen Programmierung**: Werkzeuge in Assembler (direkter Zugriff auf den CGA-Bildschirmspeicher), E-/A-Port, Erzeugung von Softwareinterrupts, Zugriff auf den gesamten PC-Speicher, Bios-Interrupt-Programmierung;
- **Tools**, u.a. Tastaturcode ermitteln, Programm-/Menügenerator, Headerdateien und Druckroutinen (Sonderzeichen, Bildschirmattribute für CGA);
- **einen C-Compiler zum Selbstprogrammieren** inkl. der wichtigsten E-/A-Funktionen. Mit der ausführlichen Beschreibung aller Einzelbaugruppen wie Token Interpreter, Parser und Code-Generatoren lernen Sie nicht nur die Arbeitsweise des Compilers verstehen, sondern erfahren auch, wie Sie individuelle Erweiterungen vornehmen;
- **den „Desktop-Menü-Conductor“**, eine komfortable Benutzerschnittstelle mit Pull-down-Menüs und ausgefeilter Windowtechnik;
- **eine Programmsammlung zur Businessgrafik** (Liniengrafik, horizontale und vertikale Balken);
- **Add-on-Lösungen zu dBase**
- **Ergänzungsausgaben zum Grundwerk** mit Erweiterungen Ihrer Anwender- und Funktionsbibliotheken, Betriebssystem- und Assemblerfunktionen sowie neuen Tools und Utilities.

Fordern Sie noch heute an:

Erfolgreiches Programmieren mit komfortablen Musterlösungen in C

stabiler Ringbuchordner, DIN A4, Grundwerk ca. 750 Seiten, inkl. Diskette, Bestell-Nr. 3600/73600, Preis: DM 98,-.

Alle 2 – 3 Monate erhalten Sie Ergänzungsausgaben zum Grundwerk mit je ca. 120 Seiten zum Seitenpreis von 38 Pfennig (Abbestellung jederzeit möglich).

Jetzt nutzen Sie Ihren PC noch effizienter



inklusive Diskette

Dieses Nachschlagewerk bietet Ihnen:

- **detailliertes Hardwarewissen** über PCs, XTs, ATs bis hin zu den neuen PS/2-Systemen. Umfassende Prozessorbeschreibungen der Intelserie 80(x)xx zeigen Ihnen die CPU-Architekturen und -Beschlagnahmen sowie die internen Abläufe bei Adressierung und Befehlsausführung;
 - **Anleitungen für den routinierten Einsatz von Systemsoftware**: Sie lernen Aufbau und Funktionen von MS-/PC-DOS und Unix kennen und erfahren, welche Möglichkeiten Ihnen die System- und Benutzerschnittstellen bieten. Sie erfahren u.a. wie Sie das Betriebssystem Ihren individuellen Wünschen anpassen können oder wie Sie durch Emulieren von IBM-BIOS-Routinen auf PC-DOS abgestimmte Programme auch auf MS-DOS-Rechnern lauffähig machen;
 - **Tuning durch Hardwareerweiterungen**, u.a. Anleitung für den Festplatteneinbau; Einsatz von Coprozessoren sowie Bauanleitungen, z.B. für eine parallele I/O-Karte oder einen universellen EPROMer;
 - **neue Tools und Utilities**: u.a. MS-DOS-Utilities, Zweifach-Tastaturtreiber, Mausanschluß für Turbo-Pascal, speicherresidente Umwandlung des Druckerodes, Batchdateien, RAM-Disk, Turbo-Pascal-Schnittstelle zu MS-DOS, V.2-Treiber, Profiler, effizientes Arbeiten mit Standardsoftware (Symphony);
 - **den kompletten Assembler SIGMASM** für hardwarenahes Programmieren;
 - **Softwareengineering**: Sie eignen sich die optimale Vorgehensweise bei der Programmentwicklung an und lernen z.B. anspruchsvolle Beschreibungsmethoden wie SADT und HIPO-Verfahren zu nutzen. Für komplexe Aufgabenstellungen erhalten Sie ausführlich dokumentierte Programmstrukturpläne. Bei der Arbeit mit Standardsoftware unterstützen Sie Templates;
 - **Musterlösungen**, z.B. zur Lagerverwaltung;
 - **Know-how für spezielle PC-Bereiche** wie DFÜ oder Netzwerke;
 - **Ergänzungsausgaben zum Grundwerk** mit neuer Anwendersoftware, weiteren Programmiersprachkursen wie C oder Lisp, aktuellen Hardware- und Betriebssystembeschreibungen (u.a. PCs mit 80386-Prozessor) sowie Tips und Utilities;
- Zusätzlicher Diskettenservice!

Fordern Sie noch heute an:

Hard- und Softwaretuning für IBM PCs und Kompatible

stabiler Ringbuchordner, DIN A 4, ca. 870 Seiten, inkl. Diskette, Bestell-Nr. 2600/72600, Preis: DM 98,-.

Alle 2 – 3 Monate erhalten Sie Ergänzungsausgaben zum Grundwerk mit je ca. 120 Seiten zum Seitenpreis von 38 Pfennig (Abbestellung jederzeit möglich).

Basic-Tools, Utilities und Anwenderprogramme für Wirtschaft, Technik, Mathematik



inklusive Diskette

Dieses Nachschlagewerk bietet Ihnen:

- auf ca. 1100 Seiten sofort einsetzbare Programme und Utilities für PC-/MS-DOS-Computer. Hinweise auf gerätespezifische Besonderheiten ermöglichen Ihnen auch die problemlose Anpassung an andere Rechner. Sie erhalten u.a.
 - **anspruchsvolle Anwenderprogramme wie**
 - die Textverarbeitung „SOTEXT 4.0“;
 - **Programme für Wirtschaft und Verwaltung**: Finanzbuchhaltung „SOBUCH“, Netzplantechnik, Terminkalender, Berechnung der Einkommenssteuer, Rabattermittlung, Tilgungsplan eines Darlehens, Restwertberechnung...;
 - **mathematische Programme**: Analyse von Funktionsgleichungen (z.B. Kurvendiskussion);
 - **statistische Programme** (u.a. Berechnung von Mittelwert, Streuung, Standardabweichung);
 - **technische Programme**: Messen/Steuern/Regeln, Wertetabellen einer einfachen Digitalschaltung, Berechnen von Querschnittswerten...;
 - **Programme für die Ausbildung** (z.B. Vokabeltraining);
 - **Tools und vielseitige Routinen für**
 - **Grafik**: Grafik-Generator „SOGRAF“, Zeichengenerator;
 - **Dateiverwaltung und Dateioorganisation**: Dateigenerator, Komprimieren und Expandieren von Textdateien, Konvertieren von Dateien, Disketten-Editor „SOZAP“, indexsequentielle Dateioorganisation und Reorganisation, Hash-Dateioorganisation; Such- und Sortierverfahren;
 - **Drucker** (z.B. Ausgabe von Random-Dateien mit Zeilenvorschub, Druckereinstellung);
 - **Bildschirm** (z.B. Bildschirm- und Menütechnik);
 - **Multitasking** (z.B. Programm zur Parallelverarbeitung);
- Jedes Programm ist ausführlich dokumentiert mit Problembeschreibung, Vorgehensweise, Struktogramm, Listing und Probelauf.

Fordern Sie noch heute an:

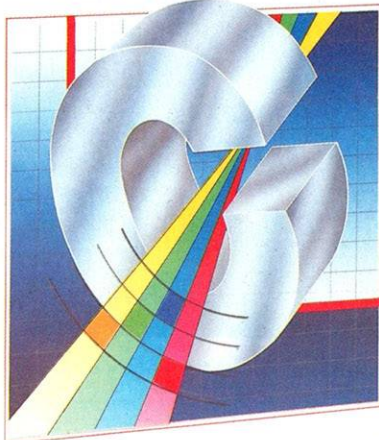
Aktuelle Musterprogramme in Basic

stabiler Ringbuchordner, DIN A 4, über 1100 Seiten, inkl. Diskette „SOTEXT 4.0“, Bestell-Nr. 1300/71300, Preis: DM 98,-.

Alle 2 – 3 Monate erhalten Sie Ergänzungsausgaben zum Grundwerk mit je ca. 120 Seiten zum Seitenpreis von 38 Pfennig (Abbestellung jederzeit möglich).

Erfolgreiches Programmieren mit komfortablen Musterlösungen in C

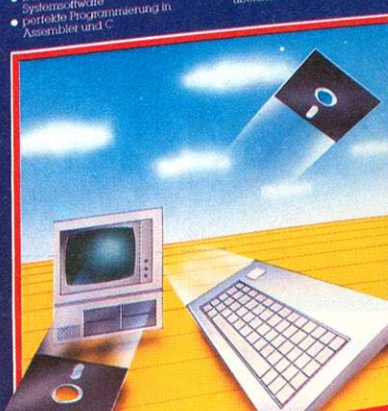
unter besonderer Berücksichtigung von Microsoft C, Lattice C und Turbo C



Hard- und Softwaretuning für IBM PCs und Kompatible

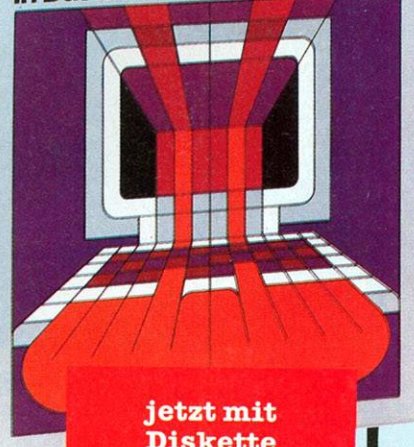
- durch detailliertes Hardwarewissen
- bessere Beherrschung von Systemsoftware
- perfekte Programmierung in Assembler und C

- neue Utilities und Musterlösungen
- zukunftsorientierten PC-Einsatz
- aktuelle Hard- und Software-übersichten



Aktuelle Musterprogramme in Basic

Lutz / Schulz-Osterloh
Praxisbuch für die erfolgreiche Programmierung wirtschaftlicher, mathematischer und technischer Programme, sowie interessanter Spiele



jetzt mit Diskette „SOTEXT 4.0“

Wer sucht, der findet

Jede Textverarbeitung kennt die Befehle »Suchen« und »Suchen und Ersetzen«. Damit lassen sich in einem Text beziehungsweise einer Datei bestimmte Zeichenfolgen (Strings) suchen und austauschen. Leider beschränkt sich die Suchfunktion immer nur auf die gerade bearbeitete Datei. »Superfind« kennt diese Einschränkung nicht: Das Pascal-Programm bearbeitet beliebig viele Dateien innerhalb eines Unterverzeichnisses.

Das Programm Superfind findet in allen Dateien des von Ihnen gewünschten Typs innerhalb eines Unterverzeichnisses einen String mit einer Länge von bis zu 255 Zeichen. Darüber hinaus darf der String gegen einen anderen ausgetauscht werden; entweder ohne oder mit Nachfrage. Zusätzlich wird auf Wunsch ein Druckerprotokoll geführt. Superfind ist benutzerfreundlich und daher einfach zu bedienen:

Nach dem Start des Programms geben Sie das gewünschte Laufwerk und den Pfad an, in dem sich die zu durchsuchenden Dateien befinden; also zum Beispiel »c:\dbase«. Anschließend bestimmen Sie die zu bearbeitenden Dateien: »*.prg«.

Superfind liest nun das Unterverzeichnis ein und stellt alle gefundenen Dateien auf dem Bildschirm dar.

Anschließend geben Sie den gesuchten String ein, er darf eine maximale Länge von 255 Zeichen aufweisen. Das Programm fragt Sie nun, ob dieser String gegen einen anderen ausgetauscht werden soll. Wenn ja, geben Sie »J« ein und anschließend den Austausch-String (er darf ebenfalls bis zu 255 Zeichen lang sein).

Sie behalten stets den Überblick über den gegenwärtigen Arbeitsablauf

Es stehen Ihnen zwei Optionen zur Verfügung: »U« bedeutet, daß die Groß- und Kleinschreibung ignoriert wird. Mit der Option »N« bestimmen Sie, daß Superfind alle gefundenen Strings gegen den Austausch-String ersetzt – und zwar ohne bei Ihnen nachzufragen. Haben Sie die Option

»N« nicht gewählt, so hält das Programm bei jeder Fundstelle an, zeigt Ihnen den Suchbegriff in seiner ursprünglichen Form und fragt, ob er ersetzt werden soll. Ist dies der Fall, geben Sie »J« ein, wird der String ersetzt und in der neuen Form noch einmal auf dem Bildschirm gezeigt. Diese Prozedur wiederholt sich so oft, wie Superfind den von Ihnen gesuchten String findet.

Im oberen Fenster wird die jeweils durchsuchte Datei angezeigt. Im rechten Teil des Fensters erscheint die gerade durchsuchte Zeile (»S«). Findet ein Austausch statt, so sehen Sie neben dem »S« noch ein »E«. Dieses zeigt die ersetzten Zeilen an. Wenn ein Austausch vollzogen wird, muß das Programm ja grundsätzlich eine Hilfsdatei eröffnen, in die es alle Zeilen, ob gefunden oder nicht, der alten Datei hineinschreibt. Die neue Datei bekommt später den Namen der alten Datei, während diese gelöscht wird. Die Prozedur verzögert zwar den Programmlauf, ist aber unumgänglich, da das Programm ja vorher nicht »wissen« kann, ob es in der alten Datei »fündig« wird.

Wenn Sie das Druckerprotokoll eingeschaltet haben, werden die entsprechenden Zeilen, in denen der gesuchte String gefunden wurde, ausgedruckt, natürlich unter Angabe der zugehörigen Datei. Hat ein Austausch stattgefunden, wird immer der neue String gedruckt.

Im Anschluß an die Suche erhalten Sie ein Protokoll, in dem die benötigte Zeit, die Anzahl der durchsuchten Dateien und Zeilen sowie die Anzahl der »Fundstellen« und Austauschvorgänge angezeigt wird (Bild2).

Wer das Programm beschleunigen möchte, kann einige Sicherheitsabfragen, beispielsweise für die Stringlänge, entfernen oder die Prozedur zum Einlesen einer Zeile aus der Datei durch ein einfaches »readln(datei,zeile)« ersetzen. Es ist allerdings grundsätzlich darauf zu achten, daß reine ASCII-Dateien verwendet werden. Bei Wordstar-Dateien kann die Zeilenlänge leicht überschritten werden. Derartige Dateien sollten deshalb vorher konvertiert werden.

Höhere Geschwindigkeit durch eingeschränkte Sicherheitsabfragen

Wer mit dBase programmiert und viele Prozedur- und Unterprogrammdateien hat, kann sich mit Superfind eventuell eine Menge Arbeit ersparen. Ändert man zum Beispiel im Hauptprogramm eine Variable, so hat man diese auch in allen Unterprogrammen zu verändern. Mußte man dazu bisher alle Dateien einzeln durchsuchen, so ist dies mit Superfind nun kein Problem mehr.

(Günther Bergdolt/zi)

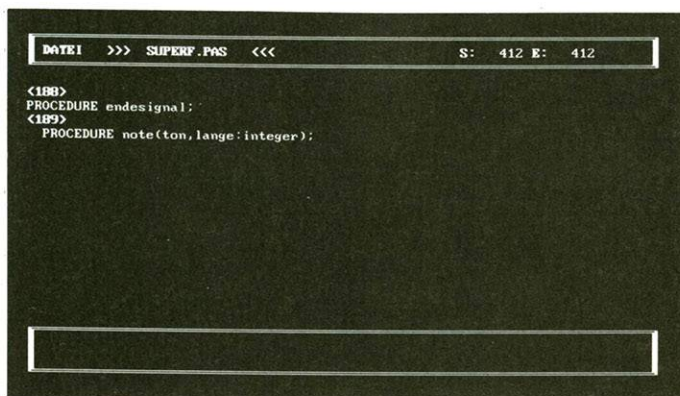


Bild 1. »Superfind« bei der Arbeit: Strings suchen und durch andere austauschen

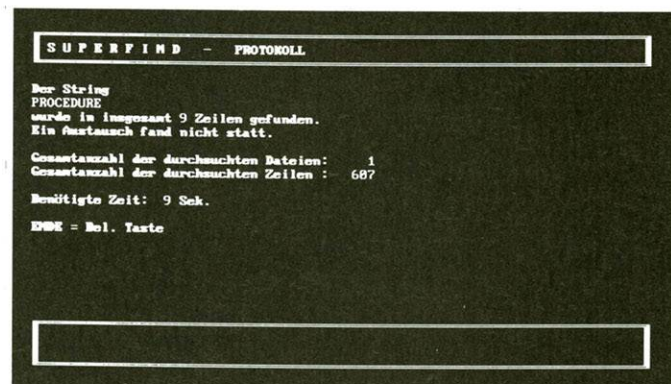


Bild 2. Mit einem Protokoll informiert Sie »Superfind« über die ausgeführten Arbeiten


```

[CP4] 1: {SR-,S+,I+,D+,T-,F-,V+,B-,N-,L+ }
[1N2] 2: {SM 16384,0,655360 }
[TU4] 3: {S+} { Stackprüfung ein }
[VS7] 4: {V-} { Keine strenge String-Überprüfung }
[LL6] 5: {SN-} {kein numerischer Coprozessor}
[UT9] 6: {SM 65500,16384,655360} {3.0-Standardvorgab
en für Stack&Heap}

[000] 7:
[FN6] 8: program superfind; { Turbo Pascal 4.0 }
[000] 9:
[7G1] 10: uses Crt,
[DR0] 11: Dos,
[LF1] 12: Printer,
[MA1] 13: Turbo3;
[000] 14:
[E14] 15: type str64 = string[64];
[UA3] 16: str255 = string[255];
[000] 17:
[8B8] 18: var dirname : array [1..5
12] of string[12];
[VS5] 19: z,anzahl,zaehler : integer;
[M36] 20: gefzaehler,ersatzzaehler : integer;
[GA4] 21: geszz : integer;
[FY4] 22: pfad,altpfad,maske : str64;
[XE4] 23: laufwerkalt : byte;
[CK5] 24: ersetzen,druck,ers,zeichen: char;
[F17] 25: suchstring,ersatzstring : string[127]

[635] 26: optionen : string[2];
[VH5] 27: ersatz,gross,frage : boolean;
[XR4] 28: datei,hilfsdatei : text;
[P14] 29: zeile : str255;
[H34] 30: position : byte;
[8F5] 31: dateiende,gefunden : boolean;
[VW4] 32: gefpos,x,y,yy : integer;
[F25] 33: ueberspringen : boolean;
[F35] 34: stdalt,minalt,sekalt : word;
[N85] 35: secl00,stunde,minu,seku : word;
[C55] 36: stdneu,minneu,sekneu : word;
[000] 37:
[036] 38: Procedure lies_dir(pfdname: str64);
[RS3] 39: var regs : registers;
[EP2] 40: x,i : integer;
[FA5] 41: fcb_bytes: array [1..43] of Byte;
[7L3] 42: eintrag : string[20];
[6L2] 43: fl : integer;
[H63] 44: ezaehler : integer;
[MU0] 45: begin
[532] 46: eintrag := '';
[E62] 47: ezaehler := 0;
[LT5] 48: for i := 1 to 43 do fcb_bytes[i] := 0;
[NG5] 49: for i := 1 to 512 do dirname[i] := '';
[HF3] 50: pfdname := pfdname + #0;
[M22] 51: with regs do begin
[CF1] 52: ax := $1A00;
[BB3] 53: ds := Seg(fcb_bytes);
[3C3] 54: dx := Ofc(fcb_bytes);
[302] 55: MSdos(regs);
[RF1] 56: ax := $4E00;
[6Z2] 57: ds := Seg(pfdname);
[533] 58: dx := Ofc(pfdname)+1;
[C41] 59: cx := 10;
[302] 60: MSdos(regs);
[2A2] 61: fl := ax and $FF;
[PY0] 62: i := 1;
[NW2] 63: if fl = 0 then begin
[P31] 64: repeat
[FV9] 65: eintrag[i] := Chr(Mem[Seg(fcb_bytes
):Ofc(fcb_bytes)+29+i]);
[8X1] 66: i := succ(i);
[BM7] 67: until not (eintrag[i-1] in [#32..#126
]) or (i > 20);
[ZN0] 68: end;
[4L3] 69: eintrag[0] := Chr(i-1);
[Y75] 70: if pos('.',eintrag) <> 0 then begin
[VM3] 71: dirname[1] := eintrag;
[G62] 72: ezaehler := 1;
[ZN0] 73: end;
[753] 74: while fl = 0 do begin
[Z20] 75: fl := 0;
[TF1] 76: ax := $4F00;
[302] 77: MSdos(regs);
[ZA2] 78: fl := ax and $FF;
[PY0] 79: i := 1;
[P31] 80: repeat
[FV9] 81: eintrag[i] := Chr(Mem[Seg(fcb_bytes
):Ofc(fcb_bytes)+29+i]);
[8X1] 82: i := succ(i);
[AS7] 83: until not (eintrag[i-1] in [#32..#126
]) or (i > 20);
[4L3] 84: eintrag[0] := Chr(i-1);
[NW2] 85: if fl = 0 then begin

```

```

[YE5] 86: if ezaehler = 0 then ezaehler := 1;
[0V3] 87: ezaehler := ezaehler + 1;
[BX4] 88: dirname[ezaehler] := eintrag;
[ZN0] 89: end;
[ZN0] 90: end;
[ZN0] 91: end;
[M53] 92: anzahl := ezaehler;
[ZN0] 93: end;
[000] 94:
[T1A] 95: procedure scanzeil(such,original:str255; su
chpos:integer);
[2P2] 96: var i : integer;
[MU0] 97: begin
[3X2] 98: gefunden := FALSE;
[AW9] 99: for i := suchpos to length(original)-leng
th(such)+1 do begin
[F18] 100: if copy(original,i,length(such)) = such
then begin
[FU1] 101: gefpos := i;
[3U2] 102: gefunden := TRUE;
[0W0] 103: exit;
[ZN0] 104: end;
[ZN0] 105: end;
[ZN0] 106: end;
[000] 107:
[KZ6] 108: function grstring(altstr:str255) : str255;
[ZV2] 109: var i : integer;
[383] 110: gstr : string[255];
[YS1] 111: gch : char;
[MU0] 112: begin
[UP1] 113: gstr := '';
[1W5] 114: for i := 1 to length(altstr) do begin
[3A4] 115: gch := upcase(altstr[i]);
[613] 116: gstr := gstr + gch;
[ZN0] 117: end;
[3Y2] 118: grstring := gstr;
[ZN0] 119: end;
[000] 120:
[0N8] 121: function vielstring(ch:str255; anzahl:byte)
: str255;
[B32] 122: var i : byte;
[ES2] 123: v : string[255];
[MU0] 124: begin
[GC3] 125: delete(v,1,length(v));
[7V4] 126: for i:=1 to anzahl do v:=v+ch;
[DG2] 127: vielstring:=v;
[ZN0] 128: end;
[000] 129:
[VP9] 130: procedure kasten(chnummer,links,rechts,oben
,unten:byte);
[B32] 131: var i : byte;
[MU0] 132: begin
[HA3] 133: gotoxy(links,oben);
[6A4] 134: if chnummer = 201 then begin
[M49] 135: write(chr(201),vielstring(chr(205),rech
ts-links-1),chr(187));
[VA5] 136: for i:=oben+1 to unten-1 do begin
[A55] 137: gotoxy(links,i); write(chr(186));
[FH5] 138: gotoxy(rechts,i); write(chr(186));
[ZN0] 139: end;
[LYC] 140: gotoxy(links,unten); write(chr(200),vie
lstring(chr(205),rechts-links-1),chr(188));
[ZN0] 141: end;
[NA4] 142: if chnummer = 218 then begin
[U49] 143: write(chr(218),vielstring(chr(196),rech
ts-links-1),chr(191));
[VA5] 144: for i:=oben+1 to unten-1 do begin
[H55] 145: gotoxy(links,i); write(chr(179));
[SH5] 146: gotoxy(rechts,i); write(chr(179));
[ZN0] 147: end;
[7ZC] 148: gotoxy(links,unten); write(chr(192),vie
lstring(chr(196),rechts-links-1),chr(177));
[ZN0] 149: end;
[DA4] 150: if chnummer = 214 then begin
[P49] 151: write(chr(214),vielstring(chr(196),rech
ts-links-1),chr(183));
[VA5] 152: for i:=oben+1 to unten-1 do begin
[A55] 153: gotoxy(links,i); write(chr(186));
[FH5] 154: gotoxy(rechts,i); write(chr(186));
[ZN0] 155: end;
[GZC] 156: gotoxy(links,unten); write(chr(211),vie
lstring(chr(196),rechts-links-1),chr(189));
[ZN0] 157: end;
[BA4] 158: if chnummer = 213 then begin
[G49] 159: write(chr(213),vielstring(chr(205),rech
ts-links-1),chr(184));
[VA5] 160: for i:=oben+1 to unten-1 do begin
[H55] 161: gotoxy(links,i); write(chr(179));
[SH5] 162: gotoxy(rechts,i); write(chr(179));
[ZN0] 163: end;
[YXC] 164: gotoxy(links,unten); write(chr(212),vie
lstring(chr(205),rechts-links-1),chr(190));
[ZN0] 165: end;

```

Listing. »superf.pas« nimmt Ihnen eintönige Arbeit ab


```

[DHD] 166:   if not ( (chnummer=201) or (chnummer=218)
[518] 167:   or (chnummer=214) or (chnummer=213) ) then
[VA5] 168:   begin
[ZA6] 169:   write(vielstring(chr(chnummer),rechts-1
[CN6] 170:   links+1));
[ZN0] 171:   for i:=oben+1 to unten-1 do begin
[EFB] 172:   gotoxy(links,i); write(chr(chnummer))
[ZN0] 173:   ;
[ZN0] 174:   gotoxy(rechts,i); write(chr(chnummer)
[ZN0] 175:   );
[3T3] 176:   end;
[3T3] 177:   gotoxy(links,unten); write(vielstring(c
[GH2] 178:   hr(chnummer),rechts-links+1));
[ZN0] 179:   end;
[000] 180:   end;
[SP3] 181:   procedure normwindow;
[000] 182:   begin
[000] 183:   window(1,1,80,25);
[000] 184:   end;
[000] 185:   end;
[Y33] 186:   procedure cloben;
[B32] 187:   var i : byte;
[MU0] 188:   begin
[MV1] 189:   gotoxy(2,2);
[VC4] 190:   for i := 1 to 53 do write(' ');
[ZN0] 191:   end;
[000] 192:   end;
[2F3] 193:   procedure clrunten;
[B32] 194:   var i : byte;
[MU0] 195:   begin
[002] 196:   gotoxy(2,23);
[7D4] 197:   for i := 1 to 77 do write(' ');
[402] 198:   gotoxy(2,24);
[7D4] 199:   for i := 1 to 77 do write(' ');
[ZN0] 200:   end;
[000] 201:   end;
[7M3] 202:   procedure endesignal;
[000] 203:   end;
[1R5] 204:   procedure note(ton,lange:integer);
[MU0] 205:   begin
[RY1] 206:   sound(ton);
[742] 207:   delay(lange);
[3G1] 208:   nosound;
[ZN0] 209:   end;
[000] 210:   end;
[MU0] 211:   begin
[PV1] 212:   note(659,60);
[412] 213:   note(880,180);
[XF1] 214:   delay(60);
[ZV1] 215:   note(880,60);
[C52] 216:   note(1108,180);
[XF1] 217:   delay(60);
[MZ1] 218:   note(1108,60);
[N52] 219:   note(1318,300);
[C52] 220:   note(1108,180);
[ZN0] 221:   end;
[000] 222:   end;
[8L5] 223:   { -----   HAUPTPROGRAMM   ----- }
[000] 224:   end;
[MU0] 225:   begin
[Y87] 226:   gefzaehler := 0; ersatzzaehler := 0; gesz
[AL5] 227:   z := 0;
[CT2] 228:   for z := 1 to 512 do dirname[z] := '';
[P22] 229:   laufwerkalt := 0;
[BL1] 230:   altpfad := '';
[000] 231:   pfad := '';
[K52] 232:   normwindow;
[DB1] 233:   clrscr;
[HV2] 234:   kasten(201,1,79,1,3);
[D63] 235:   kasten(201,1,79,22,25);
[NV1] 236:   gotoxy(3,2);
[FEA] 237:   write('S U P E R F I N D --- (C) 1988 D
OS International / G. Bergdolt');
[000] 238:   end;
[002] 239:   gotoxy(3,23);
[SY8] 240:   write('Bitte Directorypfad eingeben (z.B.
A:\TEXTE)');
[EL1] 241:   lowvideo;
[502] 242:   gotoxy(3,24);
[B12] 243:   readln(pfad);
[9U1] 244:   normvideo;
[RL1] 245:   clrunten;
[102] 246:   gotoxy(3,23);
[YP8] 247:   write('Bitte Directorymaske eingeben (z.B
*.TXT)');

```

```

[EL1] 248:   lowvideo;
[502] 249:   gotoxy(3,24);
[T82] 250:   readln(maske);
[9U1] 251:   normvideo;
[RL1] 252:   clrunten;
[6V3] 253:   pfad := grstring(pfad);
[S34] 254:   maske := grstring(maske);
[ES4] 255:   getdir(laufwerkalt,altpfad);
[Y06] 256:   if copy(pfad,length(pfad),1) = '\' then
[7G5] 257:   pfad := copy(pfad,1,length(pfad)-1);
[A31] 258:   {$I- }
[6W1] 259:   chdir(pfad);
[831] 260:   {$I+ }
[LC4] 261:   if IOresult <> 0 then begin
[YV1] 262:   gotoxy(3,5);
[A26] 263:   writeln('Directory nicht gefunden!');
[SD5] 264:   writeln('Programm abgebrochen. ');
[G55] 265:   if laufwerkalt = 0 then chdir(altpfad)
[WP1] 266:   else begin
[WAB] 267:   altpfad := chr(laufwerkalt+64)+altpfa
d;
[3G2] 268:   chdir(altpfad);
[ZN0] 269:   end;
[DU0] 270:   halt;
[ZN0] 271:   end;
[000] 272:   end;
[CN2] 273:   lies_dir(maske);
[K52] 274:   normwindow;
[7E1] 275:   cloben;
[NV1] 276:   gotoxy(3,2);
[BG8] 277:   write('Inhalt des Directorys ',pfad+chr(9
2)+maske,' ');
[F42] 278:   ausschnitt;
[DB1] 279:   clrscr;
[BE5] 280:   writeln(' (,anzahl:3, ' Dateien )');
[XF1] 281:   writeln;
[GD4] 282:   for z := 1 to anzahl do begin
[MH3] 283:   write(dirname[z]:16);
[ZN0] 284:   end;
[6M3] 285:   if anzahl = 0 then begin
[YV1] 286:   gotoxy(3,5);
[Y55] 287:   writeln('Keine Dateien gefunden!');
[SD5] 288:   writeln('Programm abgebrochen. ');
[G55] 289:   if laufwerkalt = 0 then chdir(altpfad)
[WP1] 290:   else begin
[WAB] 291:   altpfad := chr(laufwerkalt+64)+altpfa
d;
[3G2] 292:   chdir(altpfad);
[ZN0] 293:   end;
[DU0] 294:   halt;
[ZN0] 295:   end;
[K52] 296:   normwindow;
[102] 297:   gotoxy(3,23);
[V28] 298:   write('Bitte den zu suchenden String eing
eben:');
[EL1] 299:   lowvideo;
[502] 300:   gotoxy(3,24);
[V83] 301:   readln(suchstring);
[9U1] 302:   normvideo;
[RL1] 303:   clrunten;
[102] 304:   gotoxy(3,23);
[X2C] 305:   write('Soll der String durch einen andere
n String ersetzt werden (J/N) ?');
[P31] 306:   repeat
[G93] 307:   read(kbd,ersetzen);
[BY4] 308:   ersetz := upcase(ersetzen);
[PG4] 309:   until ersetz in ['J','N'];
[RL1] 310:   clrunten;
[P84] 311:   if ersetz = 'J' then begin
[102] 312:   gotoxy(3,23);
[AD6] 313:   write('Bitte Ersatzstring eingeben:');
[502] 314:   gotoxy(3,24);
[EL1] 315:   lowvideo;
[XM3] 316:   readln(ersatzstring);
[9U1] 317:   normvideo;
[YK2] 318:   ersatz := TRUE;
[4M0] 319:   end
[DG3] 320:   else ersatz := FALSE;
[RL1] 321:   clrunten;
[102] 322:   gotoxy(3,23);
[SB6] 323:   write('Bitte Optionen eingeben (U/N):');
[502] 324:   gotoxy(3,24);
[EL1] 325:   lowvideo;
[ED6] 326:   write(' U = Groß/Klein ignorieren');
[WDB] 327:   if ersatz then write(' / N = Ersetze ohne
Nachfrage');
[M42] 328:   gotoxy(35,23);
[XV2] 329:   readln(optionen);
[G55] 330:   optionen := grstring(optionen);
[LG6] 331:   if pos('U',optionen) <> 0 then gross := T
RUE
[R83] 332:   else gross := FALSE;

```



```

[WS6] 333: if pos('N',optionen) <> 0 then frage := T
RUE
[463] 334: else frage := FALSE;
[VF8] 335: if gross then suchstring := grstring(such
string);
[RL1] 336: clrnten;
[9U1] 337: normvideo;
[102] 338: gotoxy(3,23);
[GA7] 339: write('Druckerprotokoll gewünscht (J/N) ?
');
[P31] 340: repeat
[ZH2] 341: read(kbd,druck);
[VX3] 342: druck := upcase(druck);
[ST3] 343: until druck in ['J','N'];
[RM5] 344: if druck = 'J' then writeln(1st,#15);
[000] 345:
[FT5] 346: { ----- Jetzt beginnt die Arbeit ----- }
[000] 347:
[7E1] 348: clroben;
[RL1] 349: clrnten;
[000] 350:
[F42] 351: ausschnitt;
[DB1] 352: clrscr;
[AD4] 353: writeln('Bitte warten ... ');
[PS1] 354: stdalt := 0;
[HS1] 355: minalt := 0;
[SS1] 356: sekalt := 0;
[T46] 357: gettime(stdalt,minalt,sekalt,sec100);
[1F2] 358: settime(0,0,0,0);
[000] 359:
[GD4] 360: for z := 1 to anzahl do begin
[000] 361:
[UC5] 362: normwindow; clroben; clrnten;
[NV1] 363: gotoxy(3,2);
[WR7] 364: write('Scanning DATEI >>> ',dirname[
z], '<<<');
gotoxy(55,2);
write('S:
');
if ersatz then begin
gotoxy(64,2);
write('E:
');
end;
[000] 371:
[8L4] 372: assign(datei,(dirname[z]));
[K84] 373: assign(hilfsdatei,'TEMP');
[A31] 374: {$I- }
[F52] 375: reset(datei);
[831] 376: {$I+ }
[LC4] 377: if ioresult <> 0 then begin
[7S3] 378: ueberspringen := TRUE;
[4M0] 379: end
[FR4] 380: else ueberspringen := FALSE;
[000] 381:
[W65] 382: if not ueberspringen then begin
[000] 383:
[9U5] 384: if ersatz then rewrite(hilfsdatei);
[S72] 385: zzaehler := 0;
[3X2] 386: gefunden := FALSE;
[0Z2] 387: dateiende := TRUE;
[000] 388:
[7N4] 389: while not eof(datei) do begin
[000] 390:
[EM1] 391: zeile := '';
[P31] 392: repeat
[A53] 393: read(datei,zeichen);
[LT3] 394: zeile := zeile+zeichen;
[D05] 395: if length(zeile) > 254 then begin
[DB1] 396: clrscr;
[KW1] 397: write('#7#7#7);
[PE6] 398: writeln('!!! Zeilenlänge zu gr
oß !!!');
[SD5] 399: writeln('Programm abgebrochen.
');
[DU0] 400: halt;
[ZN0] 401: end;
[CY2] 402: until eoln(datei);
[Z72] 403: readln(datei);
[RM9] 404: if gross then position := pos(such
tring,grstring(zeile))
[KU6] 405: else position := pos(suchstring,zei
le);
[4P4] 406: zzaehler := succ(zzaehler);
[6H3] 407: geszz := succ(geszz);
[K52] 408: normwindow;
[EL1] 409: lowvideo;
[502] 410: gotoxy(58,2);
[X13] 411: write(zzaehler:5);
[9U1] 412: normvideo;
[F42] 413: ausschnitt;
[000] 414:
[7C4] 415: if position <> 0 then begin
[000] 416:
[ZM6] 417: write('#7); gefzaehler := succ(gef
zaehler);

```

Listing. »superf.pas« nimmt Ihnen eintönige Arbeit ab

Amstrad PC1640/Mono/Lauf.	DM 1.499,-
Amstrad LQ 3500- Drucker	DM 740,-
Copy Option Board Deluxe	DM 249,-
Copy II/PC V5.01	DM 69,-
Utah Cobol V5.0/Fortran à	DM 99,-
BTX-Term/Postmodem	DM 249,-
PC-Tools V4.3 DEUTSCH	DM 109,-
JPI-Modula	DM 179,-
Microsoft C V5.1	DM 699,-
Quick C/Quick Basic V4 à	DM 179,-
Turbo Pascal V4/C V1.5 à	DM 159,-
PC-Katalogdisk	ab DM 3,-
MS-Makroassembler V5.1	DM 289,-

Kostenlose Prospekte von

CWTG

Computerversand CWTG
Joachim Tiede,
Bergstr. 13,
D-7109 Roigheim
Tel./BTX 06298/3098
von 17-19 Uhr

Profi-Tools für QuickBASIC und Turbo BASIC

Schreiben Sie schnellere, leistungsfähigere und professionellere Programme! Wir helfen Ihnen dabei mit nützlichen Tools.

Zum Beispiel:

- Toolboxen (Fenstertechnik, Menüs, DOS-Funktionen etc.)
- Relationale Datenbank mit komfortablem Masken-Editor
- Grafik-Paket (Geschäftsgrafik und Zeichensatz-Generator)
- Maus-Programmierung
- Laserdrucker-Unterstützung

Alle Pakete mit ausführlich dokumentierten Quelltexten und Programmbeispielen. Wo erforderlich, kommen schnelle Assembler-Routinen zum Einsatz. Wollen Sie mehr aus Ihrem BASIC-Compiler herausholen? Wir informieren Sie gerne kostenlos!

**Ingenieur-Büro
Harald Zoschke**

Berliner Str. 3
D-2306 Schönberg/Holstein
Telefon 0 43 44/61 66

Eingetr. Warenzeichen:
QuickBASIC: Microsoft; Turbo BASIC: Borland

MODEM

1200 baud 199,-

Kurze PC-Karte, V21/22, 300/1200 baud, Lautspr. + RS-232 eingebaut
8039 Mikroprozessor mit 4k Control-Prg., deutsche Kurzanleitung

2400 baud 399,-

Kurze PC-Karte, V21/22/22bis, 300/1200/2400 baud, RS 232 eingeb.
COM 1-4 konfg., 89826/89827-Prozessorchips, deutsche Kurzanleitung

Technische Daten der Modems:

100% Hayes-kompatibel, Bell 183/212A und CCITT V21/22/22bis
automatische Umstellung der Baud-Raten (300/1200/2400), Selbsttests
Auto-Dial und -Answer, autom. Fehlerkorrektur, Prozess-Monitor
Halb/Voll-Duplex, Ton/Pulse-Wahl unschaltbar, Config-EEPROM
lauff. Software: ProComm, Crosstalk, SmartCom II, BitCom, OA-II usw.
in externer Ausführung auch Anschluss an ATARI, COMMODORE, APPLE
Hinweis: ohne FIZ-Münner, der Anschluss an das öffentliche Postnetz
der BRD ist nicht gestattet und kann strafrechtlich verfolgt werden.
alle Modems auch als 'EXTERNE' (+ DM 45,-)

JANUS

ELECTRONIC

Krugstr. 2
3341 Semmenstedt
Hotline: 05336- 8259
Info anfordern! ab 18 Uhr

Tel: 05336/1288, Versand per NN, alle Preise incl. NN+P/U
Wir suchen Btx-, Mailbox- und Kommunikations-Software!


```

[000] 418:
[RU3] 419:         if dateiende then begin
[K52] 420:             normwindow;
[7E1] 421:             clroben;
[RL1] 422:             clrnten;
[NV1] 423:             gotoxy(3,2);
[A16] 424:             write('DATEI   >>> ',dirname[z
], ' <<<');
[BZ5] 425:         if druck = 'N' then dateiende :
= FALSE;
[F42] 426:             ausschnitt;
[DB1] 427:             clrscr;
[G21] 428:             yy := 1;
[ZN0] 429:         end;
[F42] 430:             ausschnitt;
[Y13] 431:         if yy > 12 then begin
[DB1] 432:             clrscr;
[GV1] 433:             gotoxy(1,1);
[4M0] 434:         end
[W33] 435:         else gotoxy(1,yy);
[9U1] 436:             normvideo;
[4T3] 437:             write('<',zzaehler,'> ');
[EL1] 438:             lowvideo;
[DZ1] 439:             x := wherex;
[LZ1] 440:             y := wherex;
[EP2] 441:             writeln(zeile);
[S22] 442:             yy := wherex;
[9U1] 443:             normvideo;
[000] 444:
[H63] 445:         if ersatz then begin
[000] 446:
[SM4] 447:             gefunden := TRUE; gefpos :=1;
[CV3] 448:             while gefunden do begin
[000] 449:
[BM9] 450:                 if gross then scanzeil(suchst
ring,grstring(zeile),gefpos)
else scanzeil(suchstring,zeil
e,gefpos);
[000] 452:
[RM3] 453:                 if gefunden then begin
[000] 454:
[SR3] 455:                     if not frage then begin
[K52] 456:                         normwindow;
[RL1] 457:                         clrnten;
[L02] 458:                         gotoxy(3,23);
[F23] 459:                         write('Ersetzen (J/N) ?')
;
[P31] 460:                     repeat
[X72] 461:                         read(kbd,ers);
[I13] 462:                         ers := upcase(ers);
[7K3] 463:                         until ers in ['J','N'];
[RL1] 464:                         clrnten;
[F42] 465:                         ausschnitt;
[KG3] 466:                         if ers = 'J' then begin
[ME6] 467:                             ersatzzaehler := succ(e
rsatzzaehler);
[UP6] 468:                             delete(zeile,gefpos,len
gth(suchstring));
[A06] 469:                             insert(ersatzstring,zei
le,gefpos);
[F02] 470:                             gotoxy(x,y);
[674] 471:                             write(vielstring(' ',25
4));
[F02] 472:                             gotoxy(x,y);
[EL1] 473:                             lowvideo;
[EP2] 474:                             writeln(zeile);
[9U1] 475:                             normvideo;
[ZN0] 476:                         end;
[2G4] 477:                     end { von if not frage }
[WP1] 478:                     else begin
[ME6] 479:                         ersatzzaehler := succ(ers
atzzaehler);
[UP6] 480:                         delete(zeile,gefpos,len
gth(suchstring));
[A06] 481:                         insert(ersatzstring,zeile
,gefpos);
[F02] 482:                         gotoxy(x,y);
[674] 483:                         write(vielstring(' ',254)
);
[F02] 484:                         gotoxy(x,y);
[EL1] 485:                         lowvideo;
[EP2] 486:                         writeln(zeile);
[9U1] 487:                         normvideo;
[XG4] 488:                     end; { von if not frage }
[000] 489:
[ZN3] 490:                     gefpos := succ(gefpos);
[5H4] 491:                     end; { von if gefunden... }
[000] 492:
[F35] 493:                     end; { von while gefunden ... }
[000] 494:
[334] 495:                 end; { von if ersatz ... }
[000] 496:
[8N3] 497:             if druck = 'J' then begin
[RU3] 498:                 if dateiende then begin

```

```

[AB2] 499:                 writeln(lst);
[Z37] 500:                 writeln(lst,'DATEI >>> ',dir
name[z], ' <<<');
[7R5] 501:                 writeln(lst,'-----
-----');
[AB2] 502:                 writeln(lst);
[F43] 503:                 dateiende := FALSE;
[ZN0] 504:                 end;
[LE4] 505:                 write(lst,'<',zzaehler,'> ');
[VE3] 506:                 writeln(lst,zeile);
[ZN0] 507:                 end;
[3U2] 508:                 gefunden := TRUE;
[000] 509:
[4Z4] 510:                 end; { von if position <> 0 ... }
[000] 511:
[H63] 512:                 if ersatz then begin
[21B] 513:                     for zz := 1 to length(zeile) do w
rite(hilfsdatei,copy(zeile,zz,1));
[TE3] 514:                     writeln(hilfsdatei);
[K52] 515:                     normwindow;
[EL1] 516:                     lowvideo;
[402] 517:                     gotoxy(67,2);
[X13] 518:                     write(zzaehler:5);
[9U1] 519:                     normvideo;
[F42] 520:                     ausschnitt;
[ZN0] 521:                 end;
[000] 522:
[5T5] 523:                 end; { von while not eof(datei) ... }
[000] 524:
[642] 525:                 close(datei);
[000] 526:
[H63] 527:                 if ersatz then begin
[W13] 528:                     close(hilfsdatei);
[432] 529:                     erase(datei);
[F55] 530:                     rename(hilfsdatei,dirname[z]);
[ZN0] 531:                 end;
[000] 532:
[RM3] 533:                 if gefunden then begin
[N8E] 534:                     if druck = 'J' then writeln(lst,'--
-----');
[3X2] 535:                     gefunden := FALSE;
[ZN0] 536:                     end;
[000] 537:
[AG4] 538:                     end; { von ueberspringen }
[000] 539:
[925] 540:                     end; { von for z:=1 to anzahl }
[000] 541:
[K52] 542:                     normwindow;
[863] 543:                     clroben; clrnten;
[NV1] 544:                     gotoxy(3,2);
[LK8] 545:                     writeln('S U P E R F I N D -   PROTOKOL
L
');
[F42] 546:                     ausschnitt;
[DB1] 547:                     clrscr;
[9U1] 548:                     normvideo;
[7S3] 549:                     writeln('Der String');
[EL1] 550:                     lowvideo;
[6L3] 551:                     writeln(suchstring);
[9U1] 552:                     normvideo;
[UW8] 553:                     if gefzaehler = 0 then writeln('wurde nic
ht gefunden.')
else begin
[WP1] 554:                         write('wurde in insgesamt ');
[ZM4] 555:                         lowvideo;
[EL1] 556:                         write(gefzaehler);
[Z13] 557:                         normvideo;
[9U1] 558:                         writeln('Zeilen gefunden.');

```



```

[2Y2] 581: writeln(geszz:5);
[9U1] 582: normvideo;
[XF1] 583: writeln;
[AT1] 584: stunde := 0;
[NG1] 585: minu := 0;
[9G1] 586: seku := 0;
[TH5] 587: gettime(stunde,minu,seku,sec100);
[WS2] 588: stdneu := stunde;
[WG2] 589: minneu := minu;
[YG2] 590: sekneu := seku;
[FX4] 591: if (seku+sekalt) > 59 then begin
[7V3] 592:   seku := seku+sekalt-60;
[9N3] 593:   minu := minu+minalt+1;
[4M0] 594: end
[134] 595: else minu := minu+minalt;
[NM3] 596: if minu > 59 then begin
[9K2] 597:   minu := minu-60;
[Y64] 598:   stunde := stunde+stdalt+1
[4M0] 599: end
[AW4] 600: else stunde := stunde+stdalt;
[HK5] 601: settime(stunde,minu,seku,sec100);
[000] 602:
[FE4] 603: write('Benötigte Zeit: ');
[9N3] 604: if stdneu > 0 then begin
[EL1] 605:   lowvideo;
[2M2] 606:   write(stdneu:2);
[9U1] 607:   normvideo;
[EA2] 608:   write(' Std. ');
[2N0] 609: end;
[BN3] 610: if minneu > 0 then begin
[EL1] 611:   lowvideo;
[LL2] 612:   write(minneu:2);
[9U1] 613:   normvideo;
[9A2] 614:   write(' Min. ');
[2N0] 615: end;
[EL1] 616: lowvideo;
[1M2] 617: write(sekneu:2);
[9U1] 618: normvideo;
[4M2] 619: writeln(' Sek. ');
[000] 620:
[8N3] 621: if druck = 'J' then begin
[AB2] 622:   writeln(lst);
[GB7] 623:   writeln(lst,'S U P E R F I N D --- P
rotokoll');
[CS4] 624:   writeln(lst); writeln(lst);
[MG4] 625:   writeln(lst,'Der String');
[LB4] 626:   writeln(lst,suchstring);
[UW8] 627:   if gefzaehler = 0 then writeln('wurde n
icht gefunden. ');
[PRB] 628:   else writeln(lst,'wurde in insgesamt ',
gefzaehler, ' Zeilen gefunden. ');
[6T6] 629:   if ersatz und (ersatzzaehler > 0) then
begin
[AB2] 630:     writeln(lst);
[4H9] 631:     writeln(lst,'Er wurde dabei ',ersatzz
aehler, 'mal gegen');
[VU4] 632:     writeln(lst,ersatzstring);
[105] 633:     writeln(lst,'ausgetauscht. ');
[4M0] 634:   end
[5V7] 635:   else writeln('Ein Austausch fand nicht
statt. ');
[AB2] 636:   writeln(lst);
[PLA] 637:   writeln(lst,'Gesamtanzahl der durchsuch
ten Dateien: ',anzahl:5);
[MLA] 638:   writeln(lst,'Gesamtanzahl der durchsuch
ten Zeilen: ',geszz:5);
[Y35] 639:   write(lst,'Benötigte Zeit: ');
[9N3] 640:   if stdneu > 0 then begin
[H93] 641:     write(lst,stdneu:2);
[XZ2] 642:     write(lst,' Std. ');
[2N0] 643:   end;
[BN3] 644:   if minneu > 0 then begin
[293] 645:     write(lst,minneu:2);
[SZ2] 646:     write(lst,' Min. ');
[2N0] 647:   end;
[G93] 648:   write(lst,sekneu:2);
[HC3] 649:   writeln(lst,' Sek. ');
[T53] 650:   writeln(lst,#27#64);
[2N0] 651: end;
[602] 652: endesignal;
[XF1] 653: writeln;
[RL4] 654: writeln('ENDE = Bel. Taste');
[Y44] 655: repeat until keypressed;
[GY5] 656: if laufwerkalt = 0 then chdir(altpfad)
[WP1] 657: else begin
[ZB6] 658:   altpfad := chr(64+laufwerkalt)+altpfad;
[3G2] 659:   chdir(altpfad);
[2N0] 660: end;
[000] 661:
[KN0] 662: end.

```

Listing. »superf.pas« nimmt Ihnen eintönige Arbeit ab



TOOLBOX-Spezial

Spezial I: Turbo Prolog

Eine Toolbox mit Routinen zur "Mustererkennung", Anwendungs- und Demonstrationsbeispiele. STATIS – ein Programm zur statistischen Auswertung von Daten. NEWTON – analytisches Integrieren und Differenzieren von mathematischen Termen. dBase-Dateien als Prolog-Wissensbasis. Ein Turbo Prolog-Tracer. Handbuch zur Toolbox und Dokumentationen zu den Programmen auf der Diskette.

IBM 5 1/4 Zoll, Best.-Nr. 2801,
3 1/2 Zoll, Best.-Nr. 2802

35,- DM*

Spezial II: PasComp – Standard Pascal Compiler

Kompletter Quellcode des Compilers und des P-Code-Interpreters für Turbo Pascal 4.0 und CCD-Atari ST+. Dazu jeweils die lauffähigen Compilatoren und jetzt auch alle Texte der PASCAL-Serie PASCOMP als Dokumentation auf der Diskette!

IBM 5 1/4 Zoll, Best.-Nr. 2804,
3 1/2 Zoll, Best.-Nr. 2805

35,- DM*

Atari ST 3 1/2 Zoll, Best.-Nr. 2803

Spezial III: Echtzeitverarbeitung

Komplettes Echtzeitsystem zu Studienzwecken im Quellcode für Turbo Pascal 3.0 mit Scheduler, Debugger, Utilities und Beispielen. Anpassungen an andere Compiler, Programmierstechniken und Grundlagen der Echtzeitverarbeitung sind in einer 120-kB-Dokumentation auf Diskette beschrieben. Die ideale Ergänzung zur Serie in TOOLBOX ab Heft 12'88.

IBM 5 1/4 Zoll, Best.-Nr. 2806,
3 1/2 Zoll, Best.-Nr. 2807

35,- DM*

Spezial IV: CALC & Kurvendiskussion

Der Mathe-Compiler für Formeln und Funktionen aus PASCAL 8'87 wurde um Differenzieren und Integrieren erweitert und mit der "Kurvendiskussion" aus dem PASCAL Sonderdruck kombiniert. Komplette Quellcodes für Turbo Pascal 3.0 und Bildschirmmodule zum Kurvenzeichnen für CPM, CGA, Hercules, EGA und Apple II. Mit Dokumentation auf Disk.

IBM 5 1/4 Zoll, Best.-Nr. 2808,
3 1/2 Zoll, Best.-Nr. 2809

35,- DM*

Spezial V: STRUKTO Plus

Das beliebte Programm aus PASCAL 8'87 zur Erzeugung von Struktogrammen aus Quelltexten wird verbessert und erweitert! Es erscheint mit Dokumentation auf Disk für Turbo Pascal 3.0 Ende Dezember '88.

IBM 5 1/4 Zoll, Best.-Nr. 2811,
3 1/2 Zoll, Best.-Nr. 2812

35,- DM*

Spezial VI: Spiel & Spaß

Die besten Spiele und Knobelspässe in Farbe auf Turbo Pascal 4.0 umgesetzt! Ein Sprite-Editor für eigene grafische Ausgestaltung und dokumentierte Quellcodes für Erweiterungen sind den lauffähigen Programmen beigelegt: Stratego, Schiebung, Inverso, der Zahlenwurm...

IBM 5 1/4 Zoll, Best.-Nr. 2813,
3 1/2 Zoll, Best.-Nr. 2814

35,- DM*

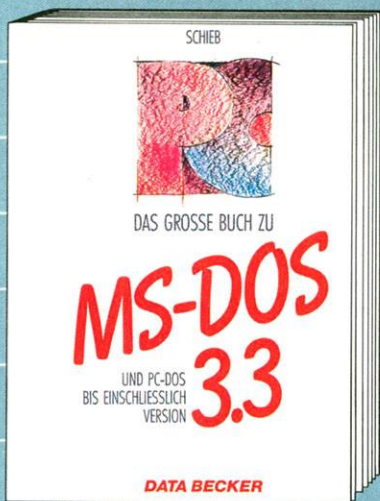
Die Spezial's III, IV und V liefern wir auf Wunsch auch für ausgesuchte CPM/Formaten und Apple II mit CPM/Karte! Genaue Informationen bietet das Merkblatt "CPM-Datobox", das Sie mit adressiertem Rückumschlag beim Softwarevertrieb des DMV-Verlages anfordern können.

* Unabhängig von der Anzahl der bestellten Artikel berechnen wir für das Inland 3,- DM bzw. für das Ausland 5,- DM Porto und Verpackung. Unverbindliche Preisempfehlung

Bitte benutzen Sie die Bestellkarte

DMV Verlag · Postfach 250 · 3440 Eschwege

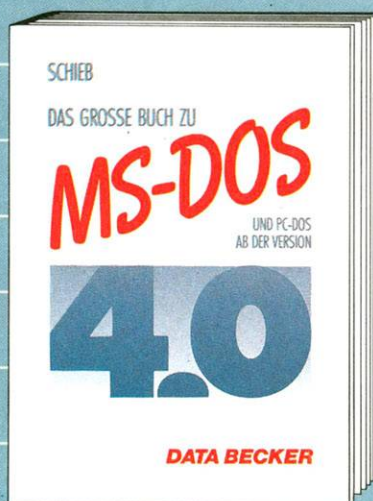
PC-Wissen, das



Komplett bis 3.3!

Das große Buch zu MS-DOS 3.3 – das bewährte Nachschlagewerk zu allen MS-DOS-Versionen bis 3.3 jetzt in der überarbeiteten Neuauflage. Mit verbesserter und erweiterter Beschreibung aller DOS-Befehle und zahlreichen Beispielen zu CONFIG.SYS- und AUTOEXEC.BAT-Dateien. So können Sie bereits vorgefertigte Angaben für die Anpassung Ihres Rechners einfach übernehmen. Bei alledem verzichtet der Autor jedoch bewußt auf unnötige Fachausdrücke und Fremdwörter, informiert aber ausführlich über die vielen wichtigen Neuerungen der 4.0-Version. Eben ein echtes Anwenderhandbuch, das voller Lösungen steckt.

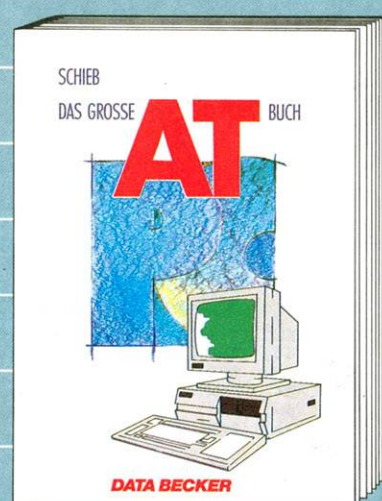
Das große Buch zu MS-DOS 3.3
Hardcover, 478 Seiten, DM 49,-



MS-DOS 4.0 – kräftig zugelegt.

Totgeglaubte leben länger: MS-DOS bleibt auch weiterhin der Standard bei Personal Computern. In der neuen Version 4.0! Ganz nach Ihren individuellen Bedürfnissen können Sie sich jetzt eine eigene „Shell“ gestalten. Wie, das zeigt Ihnen MS-DOS-Spezialist Jörg Schieb im großen Buch zu MS-DOS. Überhaupt finden Sie hier alles, was Sie zur neuen Version 4.0 wissen müssen. Von der ausführlichen Darstellung der DOS-Installation bis zur Übersicht über alle neuen Kommandos und Möglichkeiten – immer mit erklärenden Abbildungen der Menüs, der Eingabefenster oder der Hilfefunktionen. Eben das Standardwerk zum neuen DOS-Standard.

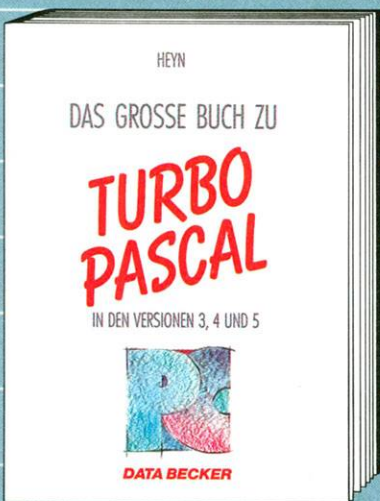
Das große Buch zu MS-/PC-DOS 4.0
Hardcover, 662 Seiten, DM 59,-



Den PC/AT verstehen.

In diesem Buch erfahren Sie alles, was Sie zum AT wissen wollen. Die Entwicklung, die Gemeinsamkeiten mit dem und die Unterschiede zum PC/XT, das Installieren des Systems, die Verwendung der Betriebssysteme DOS und OS/2, die Programmierung der Tastatur, die Organisation der Festplatte, die Unterschiede zwischen den Videokarten (MDA, CGA, VGA, PGA, Hercules), alles über EMS, LIM, Windows, Speichererweiterungen und vieles andere mehr. Mit zahlreichen Abbildungen und Programmbeispielen.

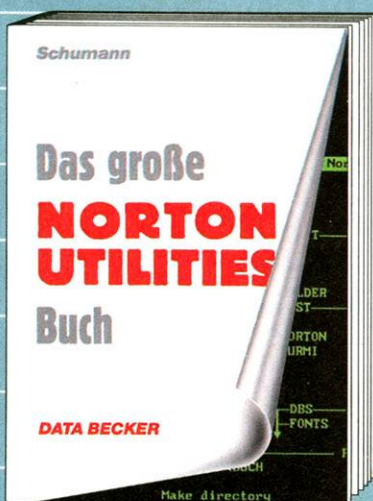
Das große AT-Buch
Hardcover, 390 Seiten, DM 69,-



Für alle Turbo-Pascal-Versionen.

Egal mit welcher Turbo-Pascal-Version Sie arbeiten – ob 3.0, 4.0 oder die aktuelle Version 5.0, das große Buch zu Turbo Pascal beschreibt Ihnen den Weg zum professionellen Turbo-Pascal-Programm: Pascal-Repetitorium, professionelle Software-Entwicklung in Turbo Pascal, Zugriffe auf DOS und BIOS, Maschinensprache einbinden, Turbo-Pascal-Intern – alle wichtigen Aspekte werden hier umfassend behandelt. Dazu bietet Ihnen dieses Buch zahlreiche Entwicklungsbausteine für eigene Programme und verrät Ihnen die Unterschiede zwischen den einzelnen Versionen.

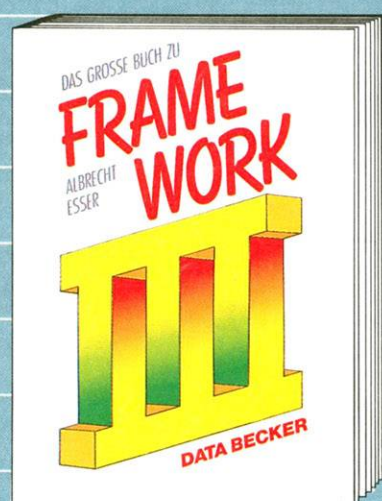
Das große Buch zu Turbo Pascal 3.0/4.0/5.0
Hardcover, ca. 800 Seiten, DM 59,-
erscheint ca. 12/88



Schnelle Hilfe zu allen Internen.

Auch ohne Vorkenntnisse die Norton-Utilities sinnvoll einsetzen? Nichts leichter als das! Mit dem Norton-Utilities-Buch. Anhand von Übungen zeigt Ihnen dieses Buch, was alles in dem Norton-Utilities-Paket steckt – beispielsweise in Sachen Datenschutz. Sowohl in den Versionen 3 und 4 als auch in der Advanced Edition. Dabei stoßen Sie nicht nur auf eine Reihe heißer Tips, sondern Sie erfahren auch noch jede Menge über die internen Vorgänge in Ihrem Rechner.

Das große Norton-Utilities-Buch
Hardcover, 317 Seiten, DM 49,-

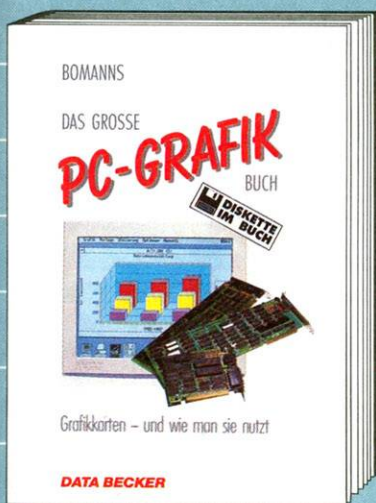


Jetzt mit Maus und neuem FRED.

Framework, bestens bekannt als Programm-Paket für die Bereiche Datenverwaltung, Tabellenkalkulation, Textverarbeitung, Präsentationsgrafik, Konzeptverwaltung und Datenfernübertragung, hat in der Version III noch einmal kräftig zugelegt. „Das große Buch zu Framework III“ beschreibt – zusammen mit den bewährten Features – ausführlich die vielen Detailverbesserungen. Einsteiger freuen sich, daß sie in der neuen Version bequem mit der Maus arbeiten können; Fortgeschrittene finden alle Neuigkeiten über FRED, die Programmiersprache von Framework.

Das große Buch zu Framework III
Hardcover, ca. 350 Seiten, DM 59,-
erscheint ca. 11/88

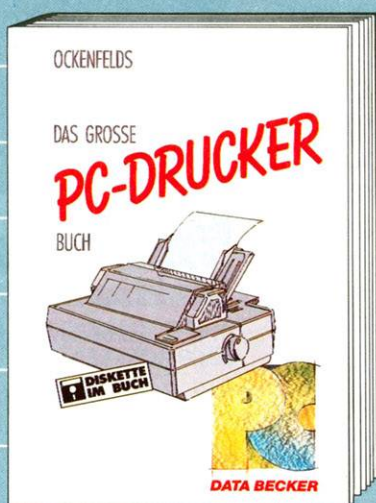
Bände spricht.



Die Welt der Grafikkarten.

CGA, HGC, HRF, EGA/AGA, VGA – welche Leistungen bieten die verschiedenen Grafikkarten? Wie werden sie installiert und wie programmiert? Welche Zusammenhänge bestehen zwischen den einzelnen Karten und den verschiedenen Programmen? Welche Verbindungen gibt es zwischen Text- und Grafikmodus? Wenn Sie umfassend informiert sein wollen, finden Sie im großen PC-Grafik-Buch die Antworten. Dabei erfahren Sie, wie Sie Zeichensätze und Grafiken editieren, Hardcopy- und Snapshot-Routinen erstellen und nutzen, Dia-Shows und Animationen programmieren und und und.

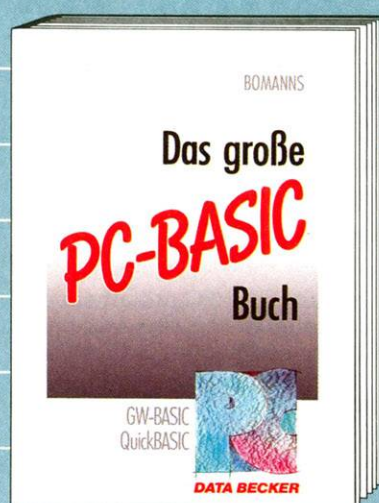
Das große PC-Grafikbuch
Hardcover, ca. 400 Seiten
inkl. Diskette DM 59,-
erscheint ca. 11/88



Rund um den PC-Drucker

Nutzen Sie Ihren Drucker richtig – lesen Sie das große PC-Drucker-Buch. Hier erfahren Sie mehr über die Bedeutung der DIP-Schalter, wie ein Druckertreiber funktioniert und was Sie tun müssen, wenn der Ausdruck einmal keine Umlaute enthält. Denn das große PC-Drucker-Buch steckt voller Lösungen. Gleichzeitig dient es als hervorragendes Nachschlagewerk bei allen Druckerproblemen und -fragen. Mit beiliegender Diskette.

Das große PC-Drucker-Buch
Hardcover, inkl. Diskette, 372 Seiten
DM 59,-



Das Anwenderhandbuch.

Das große PC-BASIC-Buch – unentbehrlich für jeden ernsthaften BASIC-Programmierer. Ein paar Stichworte aus dem Inhalt zeigen, warum: Variablen und Datentypen, Zeichenkettenverarbeitung, mathematische Funktionen und Berechnungen, Grafik und Sound, Erstellen und Einbinden von Assembler-Routinen, professioneller Umgang mit dem Editor, Fehlerbehandlung, Interrupt-Programmierung, Dateiverwaltung, Druckerausgabe und und und. Ob PC-, GW-, Quick- oder Turbo-BASIC – hier werden Sie umfassend informiert. Dazu noch eine ganze Reihe fertiger Utilities.

Das große PC-BASIC-Buch
Hardcover, 734 Seiten
inkl. Diskette DM 69,-



Das große Buch zu MS-Works. Mit Makrorecorder auf Diskette.

Machen Sie MS-Works zu Ihrem täglichen Produktionswerkzeug. Mit dem großen MS-Works-Buch. Hier lernen Sie nicht nur die einzelnen Programmteile anhand kurzer Praxisübungen kennen, sondern erfahren auch, wie ausgebildete Profis dieses Programmpaket einsetzen – mit all ihren Tips und Tricks: Problemlos automatisieren Sie sämtliche Arbeitsabläufe mittels Makros, übernehmen Grafiken in einen Text, tauschen Daten innerhalb

eines Dokuments, innerhalb eines Moduls oder auch

zwischen Modulen aus, übernehmen aus der Datenbank die entsprechenden Adressen für Serienbriefe und und und. Zahlreiche mögliche Lösungen zu so konkreten Anwendungen wie Fahrtkostenabrechnung, Terminverwaltung und Projektplanung geben Ihnen dabei nützliche Anregungen für Ihre eigenen Bedürfnisse. So ist diesem Buch auch gleich eine Diskette mit den verschiedenen Applikationen und Makros beigelegt. Ebenfalls auf Diskette: der Microsoft-Makrorecorder MSKEY.COM – ein Muß, um wirklich professionell mit MS-Works arbeiten zu können.

Das große MS-Works-Buch
Hardcover, ca. 400 Seiten
inkl. 3 1/2" und 5 1/4"-Diskette
DM 59,-
erscheint ca. 11/88

COUPON

COUPON BITTE EINSENDEN AN:

DATA BECKER
MEROWINGERSTR. 30
4000 DÜSSELDORF

Zzgl. DM 5,- Versandkosten
unabhängig von der best. Stückzahl

HIERMIT BESTELLE ICH FÜR MEINEN PC

NAME, VORNAME

STRASSE

ORT

☐ per Nachnahme ☐ Verrechnungsscheck anbei

☐ BITTE SENDEN SIE MIR KOSTENLOS UND UNVERBINDLICH IHREN NEUEN BRANDAKTUELLEN KATALOG WINTER 89 ZU.

MEIN COMPUTER:

DATA BECKER
Merowingerstr. 30 · 4000 Düsseldorf · Tel. (0211) 31 0010

Mehr als ein Blick ins CMOS-RAM

Das Programm »Cmosview« gestattet es, alle im CMOS-RAM der AT-Familie gespeicherten Daten wie Kalender oder Systemkonfiguration einzusehen, zu ändern oder den Inhalt auf Diskette zu sichern!

Zwar wird fast jeder PC mit einem eigenen Setup-Programm ausgeliefert, oft aber lassen sich mit etwas älteren Setup-Versionen längst nicht alle im AT vorgesehenen und heute erhältlichen Hardware-Optionen im CMOS-RAM definieren oder modifizieren. Neue Versionen von Setup sind später meist nur schwer erhältlich. Beispielsweise lassen sich mit vielen Setup-Versionen vor Mitte 1987 keine 3 1/2-Zoll-Laufwerke definieren.

Das Programm »Cmosview« ergänzt die Funktionen des Setup-Programms. Neben einer eigenen Setup-Routine zur Einstellung von Konfiguration und Kalender bietet Cmosview die Möglichkeit, jedes Byte im CMOS-RAM einzeln zu verändern. Das Programm speichert auf Wunsch den RAM-Inhalt auf Diskette. Falls dann einmal die Batterie ausfällt oder ein »wild gewordenes« Programm den Inhalt zerstört hat, können Sie im Nu die alten Werte im CMOS-RAM wieder herstellen.

Was ist nun dieses CMOS-RAM? Das CMOS-RAM ist ein kleiner Speicherbereich, der ständig – also auch bei ausgeschaltetem Gerät – durch die eingebaute Batterie mit Strom versorgt wird. Er ist jedoch kein Teil des normalen

Byte/Adresse (hex)	Funktion
00	Echtzeit Uhr - Sekunden
01	Echtzeit Uhr - Sekunden (Alarm)
02	Echtzeit Uhr - Minuten
03	Echtzeit Uhr - Minuten (Alarm)
04	Echtzeit Uhr - Stunden
05	Echtzeit Uhr - Stunden (Alarm)
06	Echtzeit Uhr - Wochentag
07	Echtzeit Uhr - Tag
08	Echtzeit Uhr - Monat
09	Echtzeit Uhr - Jahr
0A – 0D	Status Register (Uhr) (4 Byte)
0E	System Test Diagnose (2 Byte) wird bei
0F	Kaltstart vom POST generiert
10	Diskettenlaufwerk (A und B)
11	Reserviert (= 0)
12	Festplattenlaufwerk (Typen 1 - 14)
13	Reserviert (= 0)
14	Ausrüstung (Anzahl der Laufwerke, VDU, Coprozessor)
15	Basis-Speichergröße (unteres Byte)
16	Basis-Speichergröße (oberes Byte)
17	Erw.-Speichergröße (unteres Byte)
18	Erw.-Speichergröße (oberes Byte)
19	Disk C Ergänzungs-Byte (Typen 16-255)
1A	Disk D Ergänzungs-Byte (Typen 16-255)
1B – 1D	Reserviert (= 0)
2E – 2F	2-Byte Prüfsumme für Byte 10 bis 2D
30	Von POST ermittelte Erweiterungs-
31	Speichergröße (2 Byte)
32	Jahrhundert-Byte (= BCD 19)
33	Setup Informationsbyte
34 – 3F	Reserviert

Tabelle. Der Inhalt des CMOS-RAM Speichers.

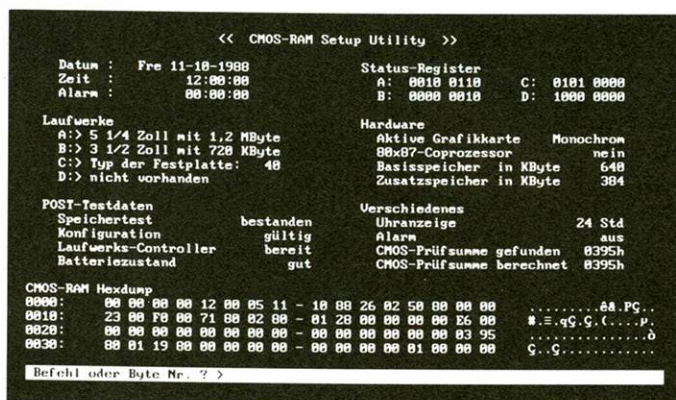


Bild 1. »cmosview« zeigt Ihnen den aktuellen Inhalt des CMOS-RAM

Arbeitsspeichers und dadurch für die meisten Programme gar nicht zugänglich. Er befindet sich in einem speziellen Chip auf der Hauptplatine des PC, dem MC146818 von Motorola. Neben dem permanenten Speicher beinhaltet dieser Chip die Echtzeituhr des PC.

Die 64 Byte (RAM) dieses Chips adressiert man in der üblichen Weise über den Port 112 (70hex), dann liest oder schreibt man über Port 113 (71hex) das gewünschte Byte. In C sieht das beispielsweise folgendermaßen aus:

```
outp(0x70, i); /* Adresse »i« an Port 112 senden */
wert = inp(0x71); /* Wert des Byte »i« aus Port 113 lesen */
outp(0x71, wert); /* über Port 113, Wert ins Byte »i« schreiben */
```

Die Tabelle gibt eine Übersicht über die Funktionen der einzelnen Bytes. Eine vollständige Beschreibung des CMOS RAM lesen Sie in der DOS, Ausgabe 7'88.

Mit der Anweisung »/H« bekommen Sie die Anleitung von »Cmosview« auf den Bildschirm

Die Statusregister (0Ahex bis 0Dhex) werden zusammen mit der Uhr vom BIOS gesetzt, und Sie sollten sie nicht verändern.

Beide Diskettenlaufwerke (A und B) sind in Byte 10hex definiert: Die oberen vier Bit definieren Laufwerk A und die unteren vier Bit das Laufwerk B.

Die Festplatten sind in ähnlicher Weise durch Byte 12hex definiert: die oberen vier Bit definieren Laufwerk C und die unteren vier Bit das Laufwerk D. Die Bezeichnungen beziehen sich auf physikalische Laufwerke und nicht auf etwaige Partitionen auf ein und derselben Platte.

Die Prüfsumme in den Byte 2E und 2Fhex ist die Summe aller Werte in den Byte 10 bis 2Dhex. Der »POST« (»Power On Self Test«) überprüft sie bei jedem Kalt- oder Warmstart.

Das Programm starten Sie einfach mit »cmosview«. Mit dem Parameter »/H« können Sie sich eine kurze Programmanleitung ausgeben lassen. Nach dem Start zeigt das Programm zunächst den aktuellen Inhalt des CMOS-RAM.

Im oberen Bereich des Bildschirms erscheinen die wichtigsten Konfigurationsdaten. Darunter sehen Sie den gesamten Speicherbereich als Hexdump, in einem von Debug her bekannten Format (Bild).

Das Programm wartet nun auf die Eingabe eines Befehls oder einer Bytenummer (in hexadezimaler Darstellung). Die Funktionen rufen Sie mit einzelnen Buchstaben auf. Die meisten Befehle beeinflussen nicht das CMOS-RAM selbst, sondern nur eine programminterne Kopie, so daß man das Programm praktisch jederzeit abbrechen kann,

ohne die CMOS-RAM-Daten zu verändern. Alle Befehle sind im folgenden aufgeführt:

- H: (Hilfe) gibt eine Zusammenfassung aller Befehle aus.
- R: (Read) liest das CMOS-RAM, speichert die Daten in einem internen Array und aktualisiert den Bildschirm.
- W: (Write) schreibt die am Bildschirm sichtbare Konfiguration (ohne Uhr und Kalender) in das CMOS-RAM.

Eine Veränderung der Konfigurationsdaten ist einzig über diesen Befehl möglich.

- S: (Sichern) speichert den Inhalt des CMOS-RAM in eine beliebige Datei. Dieser Befehl liest die Daten aus dem RAM und aktualisiert gleichzeitig die angezeigten Werte.
- L: (Laden) lädt den Inhalt einer Datei. Mit diesem Befehl wird die Datei lediglich in ein Array gelesen und am Bildschirm angezeigt. Die Daten übertragen Sie erst mit dem Befehl »write« ins CMOS-RAM.
- T: (Datum und Uhrzeit) stellen, ähnliche Funktion wie Setclock. Aktualisiert gleichzeitig den DOS-Kalender im Hauptspeicher und im CMOS-Speicher.
- D: (Disk) Laufwerkskonfiguration ändern; wirkt nur auf die programminterne Kopie des CMOS-RAM.
- Q: (Quit) Programmende, zurück zu DOS.
- [Esc]: Abbruch des laufenden Befehls.

Das CMOS-RAM wird ständig durch die eingebaute Batterie mit Strom versorgt

Geben Sie anstelle des Befehls eine Zahl (Hex 01 bis 3Fhex) ein, so erkennt das Programm diese als Byteadresse und zeigt das entsprechende Byte und dessen aktuellen Inhalt an. Den Wert des Bytes können Sie nun beliebig ändern.

Man kann bekanntlich mit Software den Computer nicht mechanisch beschädigen. Allerdings können falsche Daten im CMOS-RAM zu recht unangenehmen Problemen führen. Definieren Sie zum Beispiel keine oder die falschen Laufwerke, so können Sie den Computer nicht mehr starten. Da hilft nur noch eines: Gerät öffnen, Batterie abhängen, 15 Minuten warten und dann mit Cmosview oder Ihrem Setup-Programm die Konfiguration neu einstellen. Cmosview stellt automatisch fest (durch das Prüfen des Diagnose-Bytes 0Ehex), ob ein Setup notwendig ist.

Der Benutzer muß selbst dafür sorgen, daß die Eingaben richtig sind: Cmosview überprüft dies nicht! Falsche oder ungenügende Angaben bemerkt der POST erst beim nächsten Kalt- oder Warmstart. Um trotzdem einen gewissen Schutz vor Fehlern zu bieten, arbeitet Cmosview meist nur mit einer programminternen Kopie (Array: cmos[]). Ledig-

lich der Befehl »w« schreibt die Daten ins CMOS-RAM. Es empfiehlt sich auf jeden Fall, erst einmal eine Kopie des CMOS-RAM auf Diskette zu sichern!

Cmosview ist in Quick-C geschrieben. Es besteht aus vier Modulen. Die Datei »cmosview.c« (Listing 1) enthält das Hauptprogramm, das Setup-Programm, die Funktionen zur Änderung der Konfiguration und die Eingabe-/Ausgaberroutinen. »cmosdef.c« (Listing 2) enthält Definitionen, Includes und globale Variablen. In der Datei »cmosdis.c« (Listing 3) sind die Routinen für die Bildschirmausgabe enthalten. »cmosutil.c« (Listing 4) ist eine Sammlung von sehr hardwarenahen Funktionen und BIOS-Zugriffen.

Nachdem das Programmlisting geschrieben ist, compilieren Sie es mit:

```
QCL /AM cmosview.c
```

Am leichtesten läßt sich das Programm in der Quick-C-Umgebung (Editor) eingeben und compilieren. Dazu müssen Sie allerdings noch die in der Datei »qcsb.c« (Listing 5) enthaltenen Funktionen in die Quick-C-Library integrieren. Folgende Quick-C-Befehle erledigen dies:

```
QCL /c /AM qcsb.c
```

```
LINK quicklib.obj qcsb.obj, cmosview.qlb, , /Q;
```

Anschließend können Sie Quick-C mit dieser Quick-Library wie üblich starten:

```
QCL cmosview.c /Felcmosview
```

Cmosview leistet Ihnen sicher gute Dienste, vor allem dann, wenn Ihr Computer nicht so will, wie er soll.

(M.H. Grevé/gh/ma)

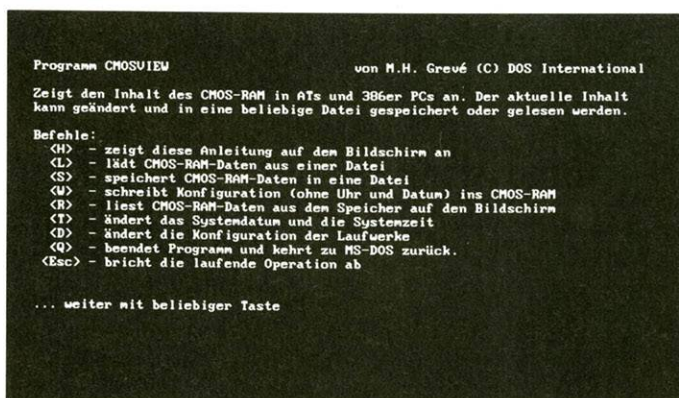
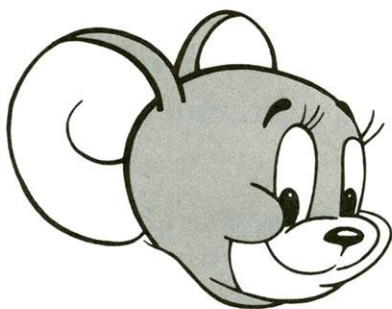


Bild 2. Das Setup-Utility »cmosview« enthält auch ein Hilfsmenu



Bringen Sie Ihre **Maus** auf Trab... nutzen Sie sie **in allen Programmen**

MausAll ist ein speicherresidentes Programm, mit dem Sie in **ALLEN** Programmen, die die Cursorstasten zur Eingabesteuerung vorsehen, die Maus einsetzen können.

MausAll emuliert aus dem Hintergrund bei allen Bewegungen der Maus in horizontaler und vertikaler Richtung die jeweilige Cursorstaste Ihrer Tastatur. Die linke und die rechte Maustaste sind mit einer Tastenkombination Ihrer Wahl belegbar.

MausAll läßt sich im Dialog oder in der Kommandozeile konfigurieren und ist somit an jedes Programm individuell anpaßbar. Durch die Arbeit von **MausAll** im Hintergrund läßt sich auch Ihr Programm-Editor (z.B. Turbo-Pascal ® 3.0-4.0), Ihre Textverarbeitung (z.B. Wordstar ®), Ihre Tabellenkalkulation (z.B. Symphony ®), oder jedes andere Programm komfortabel mit der Maus bedienen.

Für PC/XT/AT/PS-2 + Microsoft ® - Maus oder kompatible mit
MS/PC-Dos ab 2.00 und MS-kompatiblem Maustreiber (mouse.com)

MausAll...

DM 29,90

Bei Zahlung im voraus per Vr-Scheck oder
Überweisung, andernfalls Lieferung per
Nachnahme zuzüglich Versandkosten.

Csoft

Software - Entwicklungsgesellschaft - mbH
Engerstraße 188 - D-4900 Herford
Telefon (0 52 21) 3 39 30


```
[1F6] 1: /*=====
[M09] 2: ** cmosview.c -- Utility zum Untersuchen,
[R79] 3: ** und Speichern des CMOS-Speichers in der
[K60] 4: ** Echtzeituhr.
[RN7] 5: ** von M.H. Grevé, (C) 10/1988 DOS Interna
[RE6] 6: **=====
[170] 7: */
[B63] 8: #include "cmosdef.h"
[VB3] 9: #include "cmosutil.c"
[N63] 10: #include "cmosdis.c"
[000] 11:
[1F6] 12: /*=====
[GN2] 13: ** Hauptprogramm
[K60] 14: **
[3B4] 15: ** Funktions Prototypen */
[AL3] 16: void main(int, char**);
[XM3] 17: void init_scrn(void);
[UU2] 18: void setup(void);
[3B3] 19: int cmos_file(int);
[V93] 20: void kalender(void);
[7B3] 21: void laufwerk(void);
[GU3] 22: int file_name(char *);
[000] 23:
[000] 24:
[NY4] 25: void main(int argc, char **argv)
[P70] 26: {
[YU1] 27:     int cmd = 0;
[000] 28:
[878] 29:     init_scrn(); /* Bildschirm-
Initialisierung */
[8R8] 30:     if (argc) /* Parameter /
H erzeugt Hilfstext */
[Z23] 31:        strupr(argv[1]);
[9B4] 32:         if (!strcmp(argv[1], "/H")) {
[261] 33:             hilfe();
[U70] 34:         }
[U70] 35:     }
[000] 36:
[CT7] 37:     read_cmos(ALL); /* Daten aus C
MOS-RAM lesen */
[TR6] 38:     if (cmos[DIAG]) /* falls notwe
ndig, */
[416] 39:         setup(); /* Setup durch
führen */
[HL6] 40:         maske(); /* Textmaske s
chreiben */
[000] 41:
[WP5] 42:     while (1) { /* Hauptschlei
fe */
[U32] 43:         if (cmd != ESC)
[E65] 44:             showdat(); /* Daten ausge
ben */
[HR3] 45:         switch (cmd = input()) {
[AB4] 46:             case 'H': { /* Anleitung */
[261] 47:                 hilfe();
[N61] 48:                 maske();
[P01] 49:                 break;
[U70] 50:             }
[296] 51:             case 'T': { /* Datum und Z
eit ändern */
[LU1] 52:                 kalender();
[N61] 53:                 maske();
[P01] 54:                 break;
[U70] 55:             }
[FR6] 56:             case 'S': { /* Daten in Da
tei speichern */
[5R2] 57:                 cmos_file(SAVE);
[P01] 58:                 break;
[U70] 59:             }
[P56] 60:             case 'L': { /* Daten aus D
atei lesen */
[AN2] 61:                 cmos_file(LOAD);
[P01] 62:                 break;
[U70] 63:             }
[NB7] 64:             case 'W': { /* Daten ins C
MOS-RAM schreiben */
[K72] 65:                 write_cmos();
[P01] 66:                 break;
[U70] 67:             }
[G67] 68:             case 'R': { /* Daten aus d
em CMOS-RAM lesen */
[KH2] 69:                 read_cmos(ALL);
[P01] 70:                 break;
[U70] 71:             }
[N37] 72:             case 'D': { /* Laufwerkspa
rameter ändern */
[WX1] 73:                 laufwerk();
[N61] 74:                 maske();
[P01] 75:                 break;
[U70] 76:             }
[KG4] 77:             case ESC: { /* Nichts tun
*/
```

```
[P01] 78:         break;
[U70] 79:
[W24] 80:         case 'Q': { /* Programmend
e */
[Y12] 81:             cls(ncolor);
[HY7] 82:             printf("\x1b[0m"); /* ANSI-F
arben zurücksetzen */
[961] 83:             exit(0);
[U70] 84:         }
[YH6] 85:         default: { /* einzelne By
tes ändern */
[1E2] 86:             upd_byte(cmd);
[P01] 87:             break;
[U70] 88:         }
[U70] 89:     }
[U70] 90: }
[U70] 91: }
[000] 92:
[1F6] 93: /*=====
[ZNB] 94: ** Prüft auf Monochromadapter und setzt Far
bvariablen entsprechend.
[LCA] 95: ** Die Zeile mit Escape-Sequenzen kann bei
Betrieb ohne ANSI.SYS
[9D3] 96: ** weggelassen werden.
[170] 97: */
[8U2] 98: void init_scrn()
[P70] 99: {
[R77] 100:     if (video_mode() == 7) { /* Monochromad
apter */
[X52] 101:         ncolor = 0x07;
[F62] 102:         hcolor = 0x0f;
[C52] 103:         icolor = 0x70;
[D92] 104:         screen = MONO;
[U70] 105:     }
[D2A] 106:     else printf("\x1b[37m\x1b[40m"); /* ANSI
-Escape-Sequenz für */
[XX6] 107: } /* Farb
e: gelb / schwarz */
[000] 108:
[1F6] 109: /*=====
[GN2] 110: ** Setup-Routine
[170] 111: */
[4R2] 112: void setup(void)
[P70] 113: {
[B71] 114:     int i;
[000] 115:
[Y12] 116:     cls(ncolor);
[H28] 117:     printf(" CMOS-RAM muß initialisiert wer
den !");
[3H4] 118:     set_color(hcolor, -gety());
[YY3] 119:     printf("\n%s7", weiter);
[R61] 120:     getch();
[Y12] 121:     cls(ncolor);
[P07] 122:     printf(" << CMOS Setup Programm >> \n
\n");
[7H4] 123:     set_color(icolor, -gety());
[000] 124:
[YL7] 125: /* Alle Bytes der CMOS-Tabelle löschen auß
er:
[P5B] 126: ** Kalender- und Laufwerk-Bytes und Byte 30
hex sowie 31hex, welche die
[15A] 127: ** Zusatzspeichergröße enthalten. Letztere
werden vom POST
[S65] 128: ** (Power On Self Test) gesetzt.
[170] 129: */
[3B4] 130:     for (i = DIAG; i <= 0x30; ++i) {
[YG5] 131:         if (i == 0x10 || i == 0x12 || i ==
0x19
[A46] 132:             || i == 0x1a || i == 0x30 || i ==
0x31)
[AM1] 133:             continue;
[P21] 134:             cmos[i] = 0;
[U70] 135:         }
[000] 136:
[EY6] 137:         kalender(); /* Kalender /
Uhr stellen */
[785] 138:         /* Status Byte
s 0Ah - 0Dh werden */
[DP8] 139:         regs.h.ah = 7; /* von den Uhr
-Routinen gesetzt */
[SW5] 140:         int86(0x1a, &regs, &regs); /* Alarm aus
*/
[000] 141:
[E27] 142:         laufwerk(); /* Laufwerke
definieren */
[000] 143:
[CIA] 144:         int86(0x12, &regs, &regs); /* Basis-Spe
ichergröße bestimmen */
[Y54] 145:         cmos[B_MEM] = regs.h.ah;
[CH4] 146:         cmos[B_MEM + 1] = regs.h.ah;
[000] 147:
[YK8] 148:         int86(0x11, &regs, &regs); /* Ausrüstun
gs-Flag holen */
[854] 149:         cmos[EQUIP] = regs.h.ah;
[000] 150:
[NA8] 151:         regs.h.ah = 0x12; /* auf EGA/V
GA prüfen: EGA BIOS */
```



```

[XD8] 152:     regs.h.bl = 0x10;           /* aufrufen;
[K59] 153:     bei fehlender EGA- /*
[EF8] 154:     int86(0x10, &regs, &regs); /* Karte ble
[1B9] 155:     ibt BL unverändert. */
[000] 156:     if (regs.h.bl != 0x10)       /* EGA/VGA g
[KSA] 157:     efunden - Bit 4 und 5 */
[578] 158:     cmos[EQUIP] &= 0xcfc;     /* des Ausrü
[000] 159:     stungsbyte löschen */
[578] 160:     cmos[E_MEM] = cmos[0x30]; /* Zusatzspe
[ZG7] 161:     ichergröße aus Bytes */
[000] 162:     cmos[E_MEM + 1] = cmos[0x31]; /* 30 und
[578] 163:     31 holen */
[UY7] 164:     cmos[DIAG] = 0;           /* Konfigura
[000] 165:     tion Ok, Diag = 0 */
[578] 166:     checksum();                 /* Prüfsumme
[000] 167:     neu berechnen */
[578] 168:     write_cmos();               /* Setup ins
[000] 169:     CMOS-RAM schreiben */
[578] 170:     cls(hcolor);
[000] 171:     printf(" Das CMOS-RAM ist konfiguriert
[578] 172:     , führen Sie jetzt\n");
[000] 173:     printf(" einen Warmstart mit [Ctrl-Alt
[578] 174:     -Del] durch, um das\n");
[000] 175:     printf(" Betriebssystem zu aktualisier
[578] 176:     en.\n\n");
[000] 177:     printf("%s", weiter);
[578] 178:     getch();
[000] 179: }
[578] 180: }
[000] 181: }
[578] 182: }
[000] 183: }
[578] 184: }
[000] 185: }
[578] 186: }
[000] 187: }
[578] 188: }
[000] 189: }
[578] 190: }
[000] 191: }
[578] 192: }
[000] 193: }
[578] 194: }
[000] 195: }
[578] 196: }
[000] 197: }
[578] 198: }
[000] 199: }
[578] 200: }
[000] 201: }
[578] 202: }
[000] 203: }
[578] 204: }
[000] 205: }
[578] 206: }
[000] 207: }
[578] 208: }
[000] 209: }
[578] 210: }
[000] 211: }
[578] 212: }
[000] 213: }
[578] 214: }
[000] 215: }
[578] 216: }
[000] 217: }
[578] 218: }
[000] 219: }
[578] 220: }
[000] 221: }
[578] 222: }
[000] 223: }
[578] 224: }
[000] 225: }
[578] 226: }
[000] 227: }
[578] 228: }
[000] 229: }
[578] 230: }
[000] 231: }
[578] 232: }
[000] 233: }
[578] 234: }
[000] 235: }
[578] 236: }
[000] 237: }
[578] 238: }
[000] 239: }
[578] 240: }
[000] 241: }
[578] 242: }
[000] 243: }
[578] 244: }
[000] 245: }
[578] 246: }
[000] 247: }
[578] 248: }
[000] 249: }
[578] 250: }
[000] 251: }
[578] 252: }
[000] 253: }
[578] 254: }
[000] 255: }
[578] 256: }
[000] 257: }
[578] 258: }
[000] 259: }
[578] 260: }
[000] 261: }
[578] 262: }
[000] 263: }
[578] 264: }
[000] 265: }
[578] 266: }
[000] 267: }
[578] 268: }
[000] 269: }
[578] 270: }
[000] 271: }
[578] 272: }
[000] 273: }
[578] 274: }
[000] 275: }
[578] 276: }
[000] 277: }
[578] 278: }
[000] 279: }
[578] 280: }
[000] 281: }
[578] 282: }
[000] 283: }
[578] 284: }
[000] 285: }
[578] 286: }
[000] 287: }
[578] 288: }
[000] 289: }
[578] 290: }
[000] 291: }
[578] 292: }
[000] 293: }
[578] 294: }
[000] 295: }
[578] 296: }
[000] 297: }
[578] 298: }
[000] 299: }
[578] 300: }
[000] 301: }
[578] 302: }
[000] 303: }
[578] 304: }
[000] 305: }
[578] 306: }
[000] 307: }
[578] 308: }
[000] 309: }
[578] 310: }
[000] 311: }
[578] 312: }
[000] 313: }
[578] 314: }
[000] 315: }
[578] 316: }
[000] 317: }
[578] 318: }
[000] 319: }
[578] 320: }
[000] 321: }
[578] 322: }
[000] 323: }
[578] 324: }
[000] 325: }
[578] 326: }
[000] 327: }
[578] 328: }
[000] 329: }
[578] 330: }
[000] 331: }
[578] 332: }
[000] 333: }
[578] 334: }
[000] 335: }
[578] 336: }
[000] 337: }
[578] 338: }
[000] 339: }
[578] 340: }
[000] 341: }
[578] 342: }
[000] 343: }
[578] 344: }
[000] 345: }
[578] 346: }
[000] 347: }
[578] 348: }
[000] 349: }
[578] 350: }
[000] 351: }
[578] 352: }
[000] 353: }
[578] 354: }
[000] 355: }
[578] 356: }
[000] 357: }
[578] 358: }
[000] 359: }
[578] 360: }
[000] 361: }
[578] 362: }
[000] 363: }
[578] 364: }
[000] 365: }
[578] 366: }
[000] 367: }
[578] 368: }
[000] 369: }
[578] 370: }
[000] 371: }
[578] 372: }
[000] 373: }
[578] 374: }
[000] 375: }
[578] 376: }
[000] 377: }
[578] 378: }
[000] 379: }
[578] 380: }
[000] 381: }
[578] 382: }
[000] 383: }
[578] 384: }
[000] 385: }
[578] 386: }
[000] 387: }
[578] 388: }
[000] 389: }
[578] 390: }
[000] 391: }
[578] 392: }
[000] 393: }
[578] 394: }
[000] 395: }
[578] 396: }
[000] 397: }
[578] 398: }
[000] 399: }
[578] 400: }
[000] 401: }
[578] 402: }
[000] 403: }
[578] 404: }
[000] 405: }
[578] 406: }
[000] 407: }
[578] 408: }
[000] 409: }
[578] 410: }
[000] 411: }
[578] 412: }
[000] 413: }
[578] 414: }
[000] 415: }
[578] 416: }
[000] 417: }
[578] 418: }
[000] 419: }
[578] 420: }
[000] 421: }
[578] 422: }
[000] 423: }
[578] 424: }
[000] 425: }
[578] 426: }
[000] 427: }
[578] 428: }
[000] 429: }
[578] 430: }
[000] 431: }
[578] 432: }
[000] 433: }
[578] 434: }
[000] 435: }
[578] 436: }
[000] 437: }
[578] 438: }
[000] 439: }
[578] 440: }
[000] 441: }
[578] 442: }
[000] 443: }
[578] 444: }
[000] 445: }
[578] 446: }
[000] 447: }
[578] 448: }
[000] 449: }
[578] 450: }
[000] 451: }
[578] 452: }
[000] 453: }
[578] 454: }
[000] 455: }
[578] 456: }
[000] 457: }
[578] 458: }
[000] 459: }
[578] 460: }
[000] 461: }
[578] 462: }
[000] 463: }
[578] 464: }
[000] 465: }
[578] 466: }
[000] 467: }
[578] 468: }
[000] 469: }
[578] 470: }
[000] 471: }
[578] 472: }
[000] 473: }
[578] 474: }
[000] 475: }
[578] 476: }
[000] 477: }
[578] 478: }
[000] 479: }
[578] 480: }
[000] 481: }
[578] 482: }
[000] 483: }
[578] 484: }
[000] 485: }
[578] 486: }
[000] 487: }
[578] 488: }
[000] 489: }
[578] 490: }
[000] 491: }
[578] 492: }
[000] 493: }
[578] 494: }
[000] 495: }
[578] 496: }
[000] 497: }
[578] 498: }
[000] 499: }
[578] 500: }
[000] 501: }
[578] 502: }
[000] 503: }
[578] 504: }
[000] 505: }
[578] 506: }
[000] 507: }
[578] 508: }
[000] 509: }
[578] 510: }
[000] 511: }
[578] 512: }
[000] 513: }
[578] 514: }
[000] 515: }
[578] 516: }
[000] 517: }
[578] 518: }
[000] 519: }
[578] 520: }
[000] 521: }
[578] 522: }
[000] 523: }
[578] 524: }
[000] 525: }
[578] 526: }
[000] 527: }
[578] 528: }
[000] 529: }
[578] 530: }
[000] 531: }
[578] 532: }
[000] 533: }
[578] 534: }
[000] 535: }
[578] 536: }
[000] 537: }
[578] 538: }
[000] 539: }
[578] 540: }
[000] 541: }
[578] 542: }
[000] 543: }
[578] 544: }
[000] 545: }
[578] 546: }
[000] 547: }
[578] 548: }
[000] 549: }
[578] 550: }
[000] 551: }
[578] 552: }
[000] 553: }
[578] 554: }
[000] 555: }
[578] 556: }
[000] 557: }
[578] 558: }
[000] 559: }
[578] 560: }
[000] 561: }
[578] 562: }
[000] 563: }
[578] 564: }
[000] 565: }
[578] 566: }
[000] 567: }
[578] 568: }
[000] 569: }
[578] 570: }
[000] 571: }
[578] 572: }
[000] 573: }
[578] 574: }
[000] 575: }
[578] 576: }
[000] 577: }
[578] 578: }
[000] 579: }
[578] 580: }
[000] 581: }
[578] 582: }
[000] 583: }
[578] 584: }
[000] 585: }
[578] 586: }
[000] 587: }
[578] 588: }
[000] 589: }
[578] 590: }
[000] 591: }
[578] 592: }
[000] 593: }
[578] 594: }
[000] 595: }
[578] 596: }
[000] 597: }
[578] 598: }
[000] 599: }
[578] 600: }
[000] 601: }
[578] 602: }
[000] 603: }
[578] 604: }
[000] 605: }
[578] 606: }
[000] 607: }
[578] 608: }
[000] 609: }
[578] 610: }
[000] 611: }
[578] 612: }
[000] 613: }
[578] 614: }
[000] 615: }
[578] 616: }
[000] 617: }
[578] 618: }
[000] 619: }
[578] 620: }
[000] 621: }
[578] 622: }
[000] 623: }
[578] 624: }
[000] 625: }
[578] 626: }
[000] 627: }
[578] 628: }
[000] 629: }
[578] 630: }
[000] 631: }
[578] 632: }
[000] 633: }
[578] 634: }
[000] 635: }
[578] 636: }
[000] 637: }
[578] 638: }
[000] 639: }
[578] 640: }
[000] 641: }
[578] 642: }
[000] 643: }
[578] 644: }
[000] 645: }
[578] 646: }
[000] 647: }
[578] 648: }
[000] 649: }
[578] 650: }
[000] 651: }
[578] 652: }
[000] 653: }
[578] 654: }
[000] 655: }
[578] 656: }
[000] 657: }
[578] 658: }
[000] 659: }
[578] 660: }
[000] 661: }
[578] 662: }
[000] 663: }
[578] 664: }
[000] 665: }
[578] 666: }
[000] 667: }
[578] 668: }
[000] 669: }
[578] 670: }
[000] 671: }
[578] 672: }
[000] 673: }
[578] 674: }
[000] 675: }
[578] 676: }
[000] 677: }
[578] 678: }
[000] 679: }
[578] 680: }
[000] 681: }
[578] 682: }
[000] 683: }
[578] 684: }
[000] 685: }
[578] 686: }
[000] 687: }
[578] 688: }
[000] 689: }
[578] 690: }
[000] 691: }
[578] 692: }
[000] 693: }
[578] 694: }
[000] 695: }
[578] 696: }
[000] 697: }
[578] 698: }
[000] 699: }
[578] 700: }
[000] 701: }
[578] 702: }
[000] 703: }
[578] 704: }
[000] 705: }
[578] 706: }
[000] 707: }
[578] 708: }
[000] 709: }
[578] 710: }
[000] 711: }
[578] 712: }
[000] 713: }
[578] 714: }
[000] 715: }
[578] 716: }
[000] 717: }
[578] 718: }
[000] 719: }
[578] 720: }
[000] 721: }
[578] 722: }
[000] 723: }
[578] 724: }
[000] 725: }
[578] 726: }
[000] 727: }
[578] 728: }
[000] 729: }
[578] 730: }
[000] 731: }
[578] 732: }
[000] 733: }
[578] 734: }
[000] 735: }
[578] 736: }
[000] 737: }
[578] 738: }
[000] 739: }
[578] 740: }
[000] 741: }
[578] 742: }
[000] 743: }
[578] 744: }
[000] 745: }
[578] 746: }
[000] 747: }
[578] 748: }
[000] 749: }
[578] 750: }
[000] 751: }
[578] 752: }
[000] 753: }
[578] 754: }
[000] 755: }
[578] 756: }
[000] 757: }
[578] 758: }
[000] 759: }
[578] 760: }
[000] 761: }
[578] 762: }
[000] 763: }
[578] 764: }
[000] 765: }
[578] 766: }
[000] 767: }
[578] 768: }
[000] 769: }
[578] 770: }
[000] 771: }
[578] 772: }
[000] 773: }
[578] 774: }
[000] 775: }
[578] 776: }
[000] 777: }
[578] 778: }
[000] 779: }
[578] 780: }
[000] 781: }
[578] 782: }
[000] 783: }
[578] 784: }
[000] 785: }
[578] 786: }
[000] 787: }
[578] 788: }
[000] 789: }
[578] 790: }
[000] 791: }
[578] 792: }
[000] 793: }
[578] 794: }
[000] 795: }
[578] 796: }
[000] 797: }
[578] 798: }
[000] 799: }
[578] 800: }
[000] 801: }
[578] 802: }
[000] 803: }
[578] 804: }
[000] 805: }
[578] 806: }
[000] 807: }
[578] 808: }
[000] 809: }
[578] 810: }
[000] 811: }
[578] 812: }
[000] 813: }
[578] 814: }
[000] 815: }
[578] 816: }
[000] 817: }
[578] 818: }
[000] 819: }
[578] 820: }
[000] 821: }
[578] 822: }
[000] 823: }
[578] 824: }
[000] 825: }
[578] 826: }
[000] 827: }
[578] 828: }
[000] 829: }
[578] 830: }
[000] 831: }
[578] 832: }
[000] 833: }
[578] 834: }
[000] 835: }
[578] 836: }
[000] 837: }
[578] 838: }
[000] 839: }
[578] 840: }
[000] 841: }
[578] 842: }
[000] 843: }
[578] 844: }
[000] 845: }
[578] 846: }
[000] 847: }
[578] 848: }
[000] 849: }
[578] 850: }
[000] 851: }
[578] 852: }
[000] 853: }
[578] 854: }
[000] 855: }
[578] 856: }
[000] 857: }
[578] 858: }
[000] 859: }
[578] 860: }
[000] 861: }
[578] 862: }
[000] 863: }
[578] 864: }
[000] 865: }
[578] 866: }
[000] 867: }
[578] 868: }
[000] 869: }
[578] 870: }
[000] 871: }
[578] 872: }
[000] 873: }
[578] 874: }
[000] 875: }
[578] 876: }
[000] 877: }
[578] 878: }
[000] 879: }
[578] 880: }
[000] 881: }
[578] 882: }
[000] 883: }
[578] 884: }
[000] 885: }
[578] 886: }
[000] 887: }
[578] 888: }
[000] 889: }
[578] 890: }
[000] 891: }
[578] 892: }
[000] 893: }
[578] 894: }
[000] 895: }
[578] 896: }
[000] 897: }
[578] 898: }
[000] 899: }
[578] 900: }
[000] 901: }
[578] 902: }
[000] 903: }
[578] 904: }
[000] 905: }
[578] 906: }
[000] 907: }
[578] 908: }
[000] 909: }
[578] 910: }
[000] 911: }
[578] 912: }
[000] 913: }
[578] 914: }
[000] 915: }
[578] 916: }
[000] 917: }
[578] 918: }
[000] 919: }
[578] 920: }
[000] 921: }
[578] 922: }
[000] 923: }
[578] 924: }
[000] 925: }
[578] 926: }
[000] 927: }
[578] 928: }
[000] 929: }
[578] 930: }
[000] 931: }
[578] 932: }
[000] 933: }
[578] 934: }
[000] 935: }
[578] 936: }
[000] 937: }
[578] 938: }
[000] 939: }
[578] 940: }
[000] 941: }
[578] 942: }
[000] 943: }
[578] 944: }
[000] 945: }
[578] 946: }
[000] 947: }
[578] 948: }
[000] 949: }
[578] 950: }
[000] 951: }
[578] 952: }
[000] 953: }
[578] 954: }
[000] 955: }
[578] 956: }
[000] 957: }
[578] 958: }
[000] 959: }
[578] 960: }
[000] 961: }
[578] 962: }
[000] 963: }
[578] 964: }
[000] 965: }
[578] 966: }
[000] 967: }
[578] 968: }
[000] 969: }
[578] 970: }
[000] 971: }
[578] 972: }
[000] 973: }
[578] 974: }
[000] 975: }
[578] 976: }
[000] 977: }
[578] 978: }
[000] 979: }
[578] 980: }
[000] 981: }
[578] 982: }
[000] 983: }
[578] 984: }
[000] 985: }
[578] 986: }
[000] 987: }
[578] 988: }
[000] 989: }
[578] 990: }
[000] 991: }
[578] 992: }
[000] 993: }
[578] 994: }
[000] 995: }
[578] 996: }
[000] 997: }
[578] 998: }
[000] 999: }
[578] 1000: }

```

```

[K60] 223: **
[HAB] 224: ** Bei Typen >= 16 ist der entsprechende
[170] 225: ** Wert in Byte 12hex = 15 (0Fhex)
[C93] 226: void laufwerk(void)
[P70] 227: {
[6R1] 228:     int i, j;
[E92] 229:     char disktyp;
[A22] 230:     char st[81];
[000] 231:
[Y12] 232:     cls(ncolor);
[L36] 233:     printf(" << Laufwerk-Definition >> ");
[7H4] 234:     set_color(hcolor, -gety());
[DG3] 235:     for (i = 0; i < 2; ++i) {
[CV7] 236:         disktyp = (i ? cmos[0x10] & 0xf0:
[000] 237:         cmos[0x10] >> 4;
[EY5] 238:         printf("\n\nLaufwerk %c:> %s", 'A'
+ i,
[F86] 239:         floppy[(disktyp <= 5) ? dis
ktyp : 5]);
[3H4] 240:         set_color(hcolor, -gety());
[HT2] 241:         for (j = 0; j < 5; ++j)
[ST5] 242:             printf("\n\t%d.- %s", j, floppy
[j]);
[016] 243:         printf("\nNeuer Typ Code ? (0 - 15)
> ");
[CG2] 244:         if (!getstr(st))
[KC1] 245:             return;
[000] 246:         sscanf(st, "%d", &disktyp);
[WX3] 247:         cmos[0x10] = (i ? (disktyp | (cmos
[0x10] & 0xf0)) :
[000] 248:         ((disktyp << 4)
| (cmos[0x10] & 0xf0)));
[U70] 249:         for (i = 0; i < 2; ++i) {
[DG3] 250:             disktyp = (cmos[0x19 + i] ? cmos[0x
19 + i] :
[5C7] 251:             ((i ? cmos[0x12] & 0xf0 : (cmo
s[0x12] & 0xf0) >> 4);
[4VA] 252:             printf("\n\nLaufwerk %c:>\tFestplat
ten Typ: %3d", 'C' + i, disktyp);
[3H4] 253:             set_color(hcolor, -gety());
[W16] 254:             printf("\nNeuer Typ Code ? (0 - 99)
> ");
[CG2] 255:             if (!getstr(st))
[KC1] 256:                 return;
[000] 257:             sscanf(st, "%d", &disktyp);
[WX3] 258:             if (disktyp > 15) {
[C43] 259:                 cmos[0x12] |= (i ? 0xf0 : 0xf0
[000] 260:                 );
[P04] 261:                 cmos[0x19 + i] = disktyp;
[U70] 262:             }
[G51] 263:             else {
[6L4] 264:                 cmos[0x12] &= (i ? 0xf0 : 0xf0
[000] 265:                 );
[TT6] 266:                 cmos[0x12] |= (i ? disktyp : d
isktyp << 4;
[LW2] 267:                 cmos[0x19 + i] = 0;
[U70] 268:             }
[U70] 269:             checksum();
[WW1] 270:         }
[U70] 271:     }
[000] 272: }
[1F6] 273: }
[AC4] 274: ** Speichern der CMOS-Daten:
[PR9] 275: ** Datei erzeugen und Daten schreiben bezi
ehungsweise lesen
[170] 276: **
[000] 277: **
[1F4] 278: int cmos_file(int io_flag)
[P70] 279: {
[LV1] 280:     int i;
[S21] 281:     FILE *fp;
[A22] 282:     char st[81];
[R18] 283:     static char *io_fehler[] = { "Schreib"
, "Lese", };
[000] 284:
[Y58] 285:     if (!file_name(st)) return(0); /* Funkti
onsabbruch */
[CH3] 286:     if (io_flag == LOAD) {
[XW2] 287:         fp = fopen(st, "rb");
[9V5] 288:     } else
/* io_fla
g = SAVE */
[7X2] 289:         fp = fopen(st, "wb");
[000] 290:
[7Z7] 291:     if (fp == NULL) {
/* Fehler
meldung ausgeben */
[951] 292:         clr(24);
[HG5] 293:         printf("Datei %sfehler - ( Nr: %d )
%s",
[A66] 294:         io_fehler[io_flag], errno,
weiter);
[3H4] 295:         set_color(hcolor, -gety());
[6Y4] 296:         set_color(hcolor, 80 - gety());
[8Y1] 297:         printf("\n");

```

Listing 1. »cmosview.c« enthält das Hauptprogramm


```

[R61] 298:         getch();
[WT1] 299:         return(-1);
[U70] 300:     }
[000] 301:
[2R7] 302:         if (io_flag == LOAD)          /* Daten
in Array laden */
[Z43] 303:             for (i = 10; i < 64; i++)
[M64] 304:                 cmos[i] = (char)fgetc(fp);
[636] 305:             else {                    /* io_fil
e = SAVE */
[XY7] 306:                 read_cmos(ALL);        /* Daten
aus CMOS-RAM lesen */
[LZ6] 307:                 for (i = 10; i < 64; i++) /* ... un
d abspeichern */
[G33] 308:                     fputc(cmos[i],fp);
[U70] 309:             }
[000] 310:
[PT1] 311:             fclose(fp);
[U70] 312:         }
[000] 313:
[1F6] 314:         /*=====
=
[1D4] 315:         ** Eingabe eines Dateinamens
[170] 316:         */
[ZV3] 317:         int file_name(char *st)
[P70] 318:         {
[951] 319:             clr(24);
[H04] 320:             printf(" Datei Name ? >> ");
[3H4] 321:             set_color(hcolor, -gety());
[ND3] 322:             gotoxy(24, gety() + 2);
[000] 323:
[M35] 324:             if (!getstr(st) || !strlen(st))
[UP6] 325:                 return(0);          /* kein N
ame eingegeben */
[EZ6] 326:             strupr(st);              /* Großbu
chstaben */
[PS2] 327:             if (strlen(st) > 8)
[6D6] 328:                 st[8] = 0;          /* Länge
begrenzen */
[XG7] 329:             if (!strchr(st, '.')) {   /* Defaul
t-Dateiname */
[0U2] 330:                 strcat(st, ".DAT");
[FB2] 331:                 printf(".DAT");
[CK3] 332:                 set_color(icolor, -4);
[U70] 333:             }
[UL1] 334:             return(1);
[U70] 335:         }
[000] 336:
[1F6] 337:         /*=====
=
[V06] 338:         ** CMOS-RAM auslesen und beschreiben
[170] 339:         */
[EC9] 340:         void read_cmos(int stop)      /* CMOS-S
peicher auslesen */
[P70] 341:         {
[F11] 342:             int i;
[BW1] 343:             int upd = 0;
[E18] 344:             do {                      /* Wartes
chleife, falls Aufruf */
[G18] 345:                 outp(ADDR,UPD);      /* das Up
date unterbricht */
[FV4] 346:             } while (upd = inp(DATA) & 0x80);
[000] 347:
[514] 348:             for (i = 0; i < stop; ++i) {
[S42] 349:                 outp(ADDR, i);
[5D4] 350:                 cmos[i] = (char)(inp(DATA));
[U70] 351:             }
[LW8] 352:             if (cmos[6] >= 7)          /* von ma
nchen DOS-Versionen ab */
[EU7] 353:                 cmos[6] = 0;          /* DOS 3.
0 wird der Wochentag */
[618] 354:             }                          /* nicht
richtig aktualisiert */
[000] 355:
[000] 356:
[XP9] 357:         void write_cmos(void)         /* CMOS-S
peicher beschreiben */
[RU6] 358:         {                             /* (ohne
Datum und Zeit) */
[F11] 359:             int i;
[000] 360:
[1T8] 361:             for (i = 0xe; i < 64; ++i) { /* Konfig
uration schreiben */
[S42] 362:                 outp(ADDR, i);
[G93] 363:                 outp(DATA, cmos[i]);
[U70] 364:             }
[7Y7] 365:             for (i = 1; i <= 5; i +=2) { /* Alarmw
erte schreiben */
[S42] 366:                 outp(ADDR, i);
[G93] 367:                 outp(DATA, cmos[i]);
[U70] 368:             }
[U70] 369:         }
[000] 370:
[1F6] 371:         /*=====
=
[FT5] 372:         ** Einzelne Bytes im CMOS-RAM ändern
[170] 373:         */
[9E3] 374:         void upd_byte(int i)

```

```

[P70] 375: {
[A22] 376:     char st[81];
[XF2] 377:     unsigned wert;
[951] 378:     clr(24);
[ERA] 379:     printf(" Byte %2Xh: alter Wert > %2Xh <
Neuer Wert ? > ", i, cmos[i]);
[3H4] 380:     set_color(hcolor, -gety());
[ND3] 381:     gotoxy(24, gety() + 2);
[4S7] 382:     if (getstr(st)) {                /* E
ingabe als String */
[6M2] 383:         if (strlen(st)) {
[S77] 384:             sscanf(st,"%2x", &wert); /* S
tring auswerten */
[7C8] 385:             cmos[i] = (char) wert;    /* u
nd in Array speichern */
[U70] 386:         }
[U70] 387:     }
[U70] 388: }
[000] 389:
[7C6] 390: /*=====
/
[000] 391:

```

Listing 1. »cmosview.c« enthält das Hauptprogramm

```

[1F6] 1:  /*=====
=
[1X9] 2:  ** cmosdef.h - enthält alle notwendigen Def
initionen, globale
[2K8] 3:  ** Variablen, structures, union
s und includes.
[K60] 4:  **
[RN7] 5:  ** von M.H. Grevé, (C) 10/1988 DOS Interna
tional
[RE6] 6:  /*=====
=
[170] 7:  */
[L13] 8:  #include <stdio.h>
[U83] 9:  #include <stdlib.h>
[6A3] 10: #include <string.h>
[203] 11: #include <conio.h>
[VG2] 12: #include <io.h>
[9L2] 13: #include <dos.h>
[000] 14:
[L49] 15: #define PEEK(o,s) *(byte far *)(((dword)(
s) << 16) | (dword)(o))
[BVA] 16: #define POKE(o,s,b) *(byte far *)(((dword)(
s) << 16) | (dword)(o)) = (byte)(b)
[000] 17:
[KR4] 18: typedef unsigned char byte;
[ES4] 19: typedef unsigned int word;
[2X4] 20: typedef unsigned long dword;
[000] 21:
[9E7] 22: #define ADDR 0x70 /* CMOS-Portad
ressen */
[1A3] 23: #define DATA 0x71
[0Y9] 24: #define MONO 0xb000 /* MONO Bildsc
hirm Segmentadressen */
[D39] 25: #define COLOR 0xb800 /* Farbtex-Bi
ldschrirmsegment */
[M87] 26: #define SAVE 0 /* Flag: Daten
sichern */
[KP6] 27: #define LOAD 1 /* Flag: Daten
laden */
[PA7] 28: #define ESC 27 /* Tasten defi
nitionen */
[073] 29: #define BACKSP 8
[000] 30:
[MX6] 31: #define DIAG 0x0e /* Diagnostic-
Byte */
[7H7] 32: #define EQUIP 0x14 /* Ausrüstungs
byte */
[HC9] 33: #define B_MEM 0x15 /* Basis-Speic
hergrößenByte */
[EU9] 34: #define E_MEM 0x17 /* Zusatz-Spei
chergrößenbyte */
[FY5] 35: #define ALL 0x40 /* alle Bytes
*/
[F07] 36: #define UHR 0x1d /* Systemuhrby
tes */
[V97] 37: #define UPD 0x0a /* Update-Stat
usbyte */
[000] 38:

```

Listing 2. »cmosdef.c« beinhaltet die Definitionen und die Variablen

DOS-EXTRA DATABOX

– Die Software zur DOS-EXTRA Nr. 5 –
auf 2 Disketten im Format 5 1/4" oder
1 Diskette im Format 3 1/2"

sagenhaft günstig, extra viel gute Software fürs Geld!

DATABOX:

- mehr als der übliche Softwareservice
- bringt alle Programme der DOS EXTRA auf Datenträger

Für alle PCs
unter MS-DOS
als 5 1/4" - oder
3 1/2" -Diskette



Databoxinhalte der DOS EXTRA Nr. 5

- 16 Farben für die CGA-Farbgrafikkarte
- Snapshot: Bildschirmfotos per Software
- Druckerzeichensatz selbst gemacht
- mausgesteuertes Zeichenprogramm
- Editor in GW-Basic
- Setup-Programm für ATs
- Maschinensprache für GW-Basic
- Erklärtes Listing: das Basic-Spiel »Stardos«
- Neuer Checksummer für geprüfte Eingabe

2 Disketten 5 1/4"
1 Diskette 3 1/2"

(Bestell-Nr. 3813)
(Bestell-Nr. 3814)

DM 35, – *

Noch erhältlich:

Databox DOS EXTRA Nr. 1

- SAM, der DOS-Assembler
- Clock, Uhr-Utilities
- Pascal – Include-Datei zum Aufruf von BIOS-Funktionen unter Turbo Pascal
- Tips und Tricks für jeden Zweck

Diskette 5 1/4" (Bestell-Nr. 350)
Diskette 3 1/2" (Bestell-Nr. 351)

DM 24, – *

Databox DOS EXTRA Nr. 2

- DOS-Commander, eine Benutzeroberfläche für MS-DOS
- SCopy, ein professionelles Kopierprogramm
- SForth 3.1, ein Compiler im Selbstbau
- SCAD, 3D-Grafik selbstgemacht
- Hurrican, ein Actionspiel der Superlative
- Supermind, ein Strategiespiel
- Utilities, Routinen für die tägliche Arbeit
- SCheck – Checksummer für die Eingabe

4 Disketten 5 1/4" (Bestell-Nr. 352)
2 Disketten 3 1/2" (Bestell-Nr. 353)

DM 45, – *

Databox DOS EXTRA Nr. 3

- Datenfernübertragungsprogramm in 100% Assemblercode
- Herkules-Grafik-Toolbox in Turbo Pascal
- Grafikfunktionen für Turbo C
- Zeichensatzeditor für Epson-kompatible Drucker
- Treiber für AT-Tastatur
- alle begleitenden Listings zum Kurs »PC-glasklar«
- Demo für Sprachdigitalisierung
- ...und alle weiteren Listings

2 Disketten 5 1/4" (Bestell-Nr. 357)
1 Disketten 3 1/2" (Bestell-Nr. 358)

DM 35, – *

Databox DOS EXTRA Nr. 4

- Desktop-Publishing-Programm »SDTP«
- Kommandoeditor für MS-DOS
- Editor für Disketten und Festplatten
- Speicherresidente Programme auflisten und entfernen
- Paßwort-Schutz für MS-DOS
- Tastatureingabe automatisiert
- Clock-Utility für alle Grafikkarten
- Cursor und Zeichensatz für EGA-Karte programmieren
- Fenstertechnik für Turbo C
- Generator für Hilfsbildschirme
- Geprüfte Eingabe mit »SCheck«

2 Disketten 5 1/4" (Bestell-Nr. 3811)
1 Diskette 3 1/2" (Bestell-Nr. 3812)

DM 35, – *

* Unabhängig von der Anzahl der bestellten Programme berechnen wir für das Inland 3,– DM bzw. für das Ausland 5,– DM Porto und Verpackung. – Unverbindliche Preisempfehlung – Bitte benutzen Sie die Bestellkarte.

DMV-Verlag · Postfach 250 · 3440 Eschwege


```

[XF3] 39: /* Globale Variablen */
[7M2] 40: char *floppy[] =
[444] 41: { "nicht vorhanden",
[Z23] 42: "5 1/4 Zoll mit 360 KByte",
[9Z3] 43: "5 1/4 Zoll mit 1,2 MByte",
[FZ3] 44: "3 1/2 Zoll mit 720 KByte",
[PZ3] 45: "3 1/2 Zoll mit 1,44 MByte",
[NC5] 46: "unbekannter Laufwerkstyp" };
[000] 47:
[R62] 48: char *vdu[] =
[FZ1] 49: { "EGA/VGA",
[LH1] 50: "CGA/40x25",
[ZH1] 51: "CGA/80x25",
[YK2] 52: "Monochrom" };
[000] 53:
[6C9] 54: char *wtag[] = { "Son", "Mon", "Die", "Mit",
                    "Don", "Fre", "Sam" };
[MV7] 55: char *weiter = " ... weiter mit beliebiger
                    Taste ";
[US9] 56: unsigned char cmos[64]; /* Tabelle für
                    CMOS-RAM-Daten */
[WN7] 57: int errno; /* Systemfehler
                    rvariable */
[R28] 58: int screen = COLOR; /* Videosegment
                    tadresse */
[ME8] 59: char ncolor = 0x0f; /* Normale Text
                    tfarbe (gelb) */
[F99] 60: char icolor = 0xf0; /* Inversdarstellung
                    (hellgelb) */
[8W8] 61: char hcolor = 0x0c; /* Texthervorhebung
                    (hellrot) */
[000] 62:
[6B4] 63: /* Unions und Structures */
[3W2] 64: typedef struct {
[4C1] 65:     char wt;
[HD1] 66:     char tag;
[VE1] 67:     char mon;
[F91] 68:     int ja;
[E81] 69: } DATE;
[000] 70:
[3W2] 71: typedef struct {
[RF1] 72:     char std;
[3E1] 73:     char min;
[2E1] 74:     char sek;
[8A1] 75: } TIME;
[000] 76:
[CC6] 77: union REGS regs; /* register union */
[000] 78:
[7C6] 79: /*=====
[000] 80:

```

Listing 2. »cmosdef.c« beinhaltet die Definitionen und die Variablen

```

[1F6] 1: /*=====
[X17] 2: ** cmosdis.c - erzeugt die Bildschirmmaske
[507] 3: **
[3K9] 4: ** - erzeugt Programm-Anleitung
[ME9] 5: ** - holt die Werte der CMOS-RAM-Tabelle und schreibt
[K60] 6: ** diese in die entsprechende
[LN7] 7: ** Felder der Textmaske
[RE6] 8: **
[1Y1] 9: ** Textmaske
[170] 10: */
[000] 11:
[G54] 12: /* Funktionsprototypen */
[2S2] 13: void maske(void);
[2S2] 14: void hilfe(void);
[SA3] 15: void dumphex(void);
[1A3] 16: void showdat(void);
[RX3] 17: void floppy_typ(char);
[000] 18:
[000] 19:
[DX1] 20: void maske()
[P70] 21: {
[Y12] 22:     cls(ncolor);
[XW8] 23:     gotoxy(0,22); printf(" << CMOS-RAM Setup Utility >> ");

```

```

[TV4] 24:     gotoxy(2,4); printf("Datum :");
[9S4] 25:     gotoxy(3,4); printf("Zeit :");
[BU4] 26:     gotoxy(4,4); printf("Alarm :");
[000] 27:
[NH6] 28:     gotoxy(2,42); printf("Status-Register");
[E54] 29:     gotoxy(3,44); printf("A:");
[M54] 30:     gotoxy(3,62); printf("C:");
[L54] 31:     gotoxy(4,44); printf("B:");
[T54] 32:     gotoxy(4,62); printf("D:");
[000] 33:
[RA5] 34:     gotoxy(6,2); printf("Laufwerke");
[S55] 35:     gotoxy(6,42); printf("Hardware");
[W04] 36:     gotoxy(7,4); printf("A:>");
[2S6] 37:     gotoxy(7,44); printf("Aktive Grafikkarte");
[Y04] 38:     gotoxy(8,4); printf("B:>");
[LU6] 39:     gotoxy(8,44); printf("80x87-Coprozessor");
[014] 40:     gotoxy(9,4); printf("C:>");
[NF7] 41:     gotoxy(9,44); printf("Basisspeicher in KByte");
[T24] 42:     gotoxy(10,4); printf("D:>");
[NP7] 43:     gotoxy(10,44); printf("Zusatzspeicher in KByte");
[000] 44:
[S76] 45:     gotoxy(12,2); printf("POST-Testdaten");
[856] 46:     gotoxy(12,42); printf("Verschiedenes");
[2Z5] 47:     gotoxy(13,4); printf("Speichertest");
[8F5] 48:     gotoxy(13,44); printf("Uhranzeige");
[A66] 49:     gotoxy(14,4); printf("Konfiguration");
[W44] 50:     gotoxy(14,44); printf("Alarm");
[CD7] 51:     gotoxy(15,4); printf("Laufwerks-Control");
[W48] 52:     gotoxy(15,44); printf("CMOS-Prüfsumme gefunden");
[HE6] 53:     gotoxy(16,4); printf("Batteriezustand");
[FC8] 54:     gotoxy(16,44); printf("CMOS-Prüfsumme berechnet");
[U70] 55: }
[000] 56:
[1F6] 57: /*=====
[2C3] 58: ** Programmanleitung
[170] 59: */
[DX1] 60: void hilfe()
[P70] 61: {
[Y12] 62:     cls(ncolor);
[DV1] 63:     gotoxy(1,0);
[MVC] 64:     printf("\n Programm CMOSVIEW\t\t\t von M. H. Grevé (C) DOS International\n");
[1G9] 65:     " Zeigt den Inhalt des CMOS-RAM in ATs und 386er PCs an. Der"
[483] 66:     " aktuelle Inhalt\n"
[67A] 67:     " kann geändert und in eine beliebige Datei gespeichert oder"
[HU3] 68:     " gelesen werden.\n\n");
[XA3] 69:     printf("<H> - zeigt diese Anleitung auf dem Bildschirm an\n");
[FV8] 70:     "<L> - lädt CMOS-RAM-Daten aus einer Datei\n";
[8G7] 71:     "<S> - speichert CMOS-RAM-Daten in eine Datei\n";
[C48] 72:     "<W> - schreibt Konfiguration (ohne Uhr und Datum) ins CMOS-RAM\n";
[B4B] 73:     "<R> - liest CMOS-RAM-Daten aus dem Speicher auf den Bildschirm\n";
[05B] 74:     "<T> - ändert das Systemdatum und die Systemzeit\n";
[YR8] 75:     "<D> - ändert die Konfiguration der Laufwerke\n";
[DB8] 76:     "<Q> - beendet Programm und kehrt zu MS-DOS zurück.\n";
[A99] 77:     "<Esc> - bricht die laufende Operation ab\n\n");
[UY7] 78:     printf("%s\n", weiter);
[9R3] 79:     getch();
[R61] 80: }
[U70] 81: }
[000] 82:
[1F6] 83: /*=====
[0P3] 84: ** Hexdump des CMOS-RAM
[170] 85: */
[XA2] 86: void dumphex()
[P70] 87: {
[5N1] 88:     int i = 0;
[E83] 89:     char unsigned byte;
[PA3] 90:     char asciidump[17];
[000] 91:
[FY6] 92:     gotoxy(18,0); printf("CMOS-RAM Hexdump\n");
[SU7] 93:     asciidump[16] = 0; /* String-Ende markieren */
[H22] 94:     while (i < 64) {
[1N2] 95:         byte = cmos[i];
[1R6] 96:         asciidump[i % 16] = (byte > 32) ? byte : '.';
[3M1] 97:         if (i % 16)

```



```

[703] 98:      printf("%04X: ", i);
[V63] 99:      printf("%02X ", byte);
[W92] 100:     if ( (++i % 16) == 0)
[2C4] 101:         printf("\n", asciidump);
[W82] 102:     if ( (i % 16) == 8)
[5T1] 103:         printf("- ");
[U70] 104:     }
[U70] 105: }
[000] 106:
[1F6] 107: /*=====
[135] 108: ** 1 Byte im Bin-Format ausgeben
[170] 109: */
[PM5] 110: void printbin(unsigned char byte)
[P70] 111: {
[V51] 112:     char i;
[XE3] 113:     for (i = 0; i < 8; i++) {
[YA3] 114:         if (!(i % 4)) printf(" ");
[615] 115:         putchar((byte & (0x80 >> i)) ? '1' : '0'
);
[U70] 116:     }
[U70] 117: }
[000] 118:
[1F6] 119: /*=====
[MU5] 120: ** Daten in Bildschirmmaske schreiben
[170] 121: */
[6A2] 122: void showdat()
[P70] 123: {
[UG1] 124:     char typ;
[F11] 125:     int i;
[622] 126:     int chksum;
[W12] 127:     char tag[4];
[000] 128:
[R04] 129:     /* Datum, Zeit und Alarm */
[6S4] 130:     if (cmos[6] >= 0 && cmos[6] < 7)
[MH4] 131:         strcpy(tag, wtag[cmos[6]]);
[G51] 132:     else {
[MC3] 133:         itoa(cmos[6], tag, 16);
[FC2] 134:         strcat(tag, " ");
[U70] 135:     }
[102] 136:     gotoxy(2, 14);
[DBB] 137:     printf("%s %02x-%02x-%02x%02x ", tag, cm
os[7], cmos[8], cmos[50], cmos[9]);
[7DA] 138:     gotoxy(3, 20); printf("%02x:%02x:%02x ",
cmos[4], cmos[2], cmos[0]);
[GDA] 139:     gotoxy(4, 20); printf("%02x:%02x:%02x ",
cmos[5], cmos[3], cmos[1]);
[000] 140:
[H53] 141:     /* Statusregister */
[ET5] 142:     gotoxy(3, 47); printbin(cmos[0xa]);
[GT5] 143:     gotoxy(4, 47); printbin(cmos[0xb]);
[ET5] 144:     gotoxy(3, 65); printbin(cmos[0xc]);
[GT5] 145:     gotoxy(4, 65); printbin(cmos[0xd]);
[000] 146:
[SG3] 147:     /* Floppy-Laufwerke */
[413] 148:     for(i = 0; i < 2; ++i) {
[X68] 149:         typ = (i ? cmos[0x10] & 0xf : (cmos
[0x10] & 0xf) >> 4);
[N82] 150:         gotoxy(i + 7, 8);
[EW2] 151:         floppy_typ(typ);
[U70] 152:     }
[000] 153:
[KU2] 154:     /* Festplatten */
[413] 155:     for(i = 0; i < 2; ++i) {
[R06] 156:         typ = (cmos[0x19 + i] ? cmos[0x19 + i
] :
[5C7] 157:             ((i ? cmos[0x12] & 0xf : (cm
os[0x12] & 0xf) >> 4);
[FB2] 158:             gotoxy(i + 9, 8);
[891] 159:             if (typ)
[8A6] 160:                 printf("Typ der Festplatte: %3d", t
yp);
[ZR0] 161:             else
[245] 162:                 printf("nicht vorhanden ");
[U70] 163:         }
[000] 164:
[117] 165:     /* VDU, Coprozessor und Speichergröße */
[FZ8] 166:     gotoxy(7, 66); printf("%s", vdu[(cmos[EQU
IP] & 0x30) >> 4]);
[0H9] 167:     gotoxy(8, 70); printf("%s", (cmos[EQUIP]
& 2) ? "vorh." : "nein");
[WWA] 168:     gotoxy(9, 70); printf("%5d", (int) (cmos[
B_MEM] + (cmos[B_MEM + 1] << 8)));
[MYA] 169:     gotoxy(10, 70); printf("%5d", (int) (cmos[
E_MEM] + (cmos[E_MEM + 1] << 8)));
[000] 170:
[2C3] 171:     /* POST-Auswertung */
[ZHA] 172:     gotoxy(13, 27); printf("%s", (cmos[DIAG] &
16) ? "fehler" : "bestanden");
[EX9] 173:     gotoxy(14, 30); printf("%s", (cmos[DIAG] &
32) ? "fehler" : "gültig");
[LP9] 174:     gotoxy(15, 30); printf("%s", (cmos[DIAG] &
8) ? "fehler" : "bereit");
[YZA] 175:     gotoxy(16, 30); printf("%s", (cmos[0xd] &
cmos[DIAG]) ? "fehler" : "gut");
[000] 176:
[0A2] 177:     /* Zeitmodus */

```

```

[589] 178:     gotoxy(13, 69); printf("%s", (cmos[0xb] &
2) ? "24 Std" : "12 Std");
[ZD8] 179:     gotoxy(14, 72); printf("%s", (cmos[0xb] &
32) ? "ein" : "aus");
[000] 180:
[W92] 181:     /* Checksumme */
[DV5] 182:     for (chksum = 0, i = 0x10; i <= 0x2d; ++
i)
[303] 183:         chksum += cmos[i];
[PDA] 184:     gotoxy(15, 70); printf("%04Xh", (int) ((cm
os[0x2e] << 8) + cmos[0x2f]));
[4T5] 185:     gotoxy(16, 70); printf("%04Xh", chksum);
[000] 186:
[NR1] 187:     dumphex();
[U70] 188: }
[000] 189:
[1F6] 190: /*=====
[X44] 191: ** Floppy-Laufwerkstypen
[170] 192: */
[L34] 193: void floppy_typ(char i)
[P70] 194: {
[3L5] 195:     printf("%s", floppy[(i <= 5) ? i : 5]);
[MF1] 196:     if (i > 4) {
[W03] 197:         gotoxy(getx(), 31);
[SK2] 198:         printf("%3d", i);
[U70] 199:     }
[U70] 200: }
[000] 201:
[7C6] 202: /*=====
[000] 203:

```

Listing 3. »cmosdis.c« ist für die Bildschirmausgabe zuständig

```

[1F6] 1: /*=====
[TD8] 2: ** cmosutil.c - Untergeordnete Funktionen u
nd BIOS-
[8V5] 3: ** Zugriffe für CMOSVIEW-Utility.
[K60] 4: **
[RN7] 5: ** von M.H. Grevé, (C) 10/1988 DOS Interna
tional
[RE6] 6: **=====
[170] 7: */
[000] 8:
[G54] 9: /* Funktionsprototypen */
[0W3] 10: int video_mode(void);
[ZX3] 11: void put_ch(char, char);
[RE3] 12: void get_date(DATE *);
[BS3] 13: char wochentag(DATE *);
[UK3] 14: void get_time(TIME *);
[NM4] 15: char set_time(TIME *, TIME *);
[TM3] 16: void write_kal(void);
[C63] 17: int getstr(char *);
[1C3] 18: void read_cmos(int);
[C04] 19: void write_cmos(void);
[3C3] 20: void upd_byte(int);
[000] 21:
[1F6] 22: /*=====
[K73] 23: ** Cursorzeile holen
[170] 24: */
[BE2] 25: int getx(void)
[P70] 26: {
[A12] 27:     regs.h.ah = 3;
[MS3] 28:     int86(0x10, &regs, &regs);
[R23] 29:     return (regs.h.dh);
[U70] 30: }
[000] 31:
[1F6] 32: /*=====
[4K3] 33: ** Cursorspalte holen
[170] 34: */
[CE2] 35: int gety(void)
[P70] 36: {
[A12] 37:     regs.h.ah = 3;
[MS3] 38:     int86(0x10, &regs, &regs);
[Z23] 39:     return (regs.h.dl);
[U70] 40: }
[000] 41:
[1F6] 42: /*=====
[0V3] 43: ** Cursor auf x,y setzen

```

Listing 4. In »cmosutil.c« befinden sich die BIOS-Zugriffe


```

[170] 44: */
[444] 45: void gotoxy(int x, int y)
[P70] 46: {
[512] 47:     regs.h.bh = 0;
[812] 48:     regs.h.ah = 2;
[A23] 49:     regs.h.dh = (char)x;
[P23] 50:     regs.h.dl = (char)y;
[MS3] 51:     int86(0x10, &regs, &regs);
[U70] 52: }
[000] 53:
[1F6] 54: /*=====
=
[ZPB] 55: ** Bildschirm löschen (verwendet Funktion
07 des Video-Interrupt 10hex)
[170] 56: */
[W33] 57: void cls(char color)
[P70] 58: {
[K12] 59:     regs.h.ah = 7;
[C12] 60:     regs.h.al = 0;
[VU2] 61:     regs.h.bh = color;
[P32] 62:     regs.x.cx = 0;
[282] 63:     regs.h.dh = 24;
[392] 64:     regs.h.dl = 79;
[MS3] 65:     int86(0x10, &regs, &regs);
[CV1] 66:     gotoxy(0,0);
[U70] 67: }
[000] 68:
[1F6] 69: /*=====
=
[F28] 70: ** Zeile löschen und aktuelle Farbe beibeha
lten
[170] 71: */
[DG2] 72: void cln(int n)
[P70] 73: {
[U11] 74:     gotoxy(n, 0);
[UD2] 75:     regs.h.ah = 0xa;
[052] 76:     regs.h.al = ' ';
[HA2] 77:     regs.x.cx = 80;
[512] 78:     regs.h.bh = 0;
[XN3] 79:     int86(0x10, &regs, &regs);
[U70] 80: }
[000] 81:
[1F6] 82: /*=====
=
[MN5] 83: ** Aktuellen Videomodus bestimmen
[170] 84: */
[5G3] 85: int video_mode(void)
[P70] 86: {
[082] 87:     regs.h.ah = 15;
[MS3] 88:     int86(0x10, &regs, &regs);
[W23] 89:     return (regs.h.al);
[U70] 90: }
[000] 91:
[1F6] 92: /*=====
=
[WW6] 93: ** Zeichen mit Farbe ohne Vorschub ausgeben
[170] 94: */
[GB5] 95: void put_ch(char ch, char color)
[P70] 96: {
[P12] 97:     regs.h.ah = 9;
[5B2] 98:     regs.h.al = ch;
[6T4] 99:     regs.h.bl = (color) ? color : 7;
[512] 100:     regs.h.bh = 0;
[S32] 101:     regs.x.cx = 1;
[MS3] 102:     int86(0x10, &regs, &regs);
[U70] 103: }
[000] 104:
[1F6] 105: /*=====
=
[T98] 106: ** Zeichen mit Vorschub in aktueller Farbe
ausgeben
[170] 107: */
[FN3] 108: void write_ch(char ch)
[P70] 109: {
[X72] 110:     regs.h.ah = 14;
[512] 111:     regs.h.bh = 0;
[5B2] 112:     regs.h.al = ch;
[MS3] 113:     int86(0x10, &regs, &regs);
[U70] 114: }
[000] 115:
[1F6] 116: /*=====
=
[8L6] 117: ** Farbe eines Strings der Länge l ändern.
[VTA] 118: ** Die Routine beginnt mit der aktuellen Cu
rsorposition und über
rsorposition und über
[X2B] 119: ** schreibt das Farbattribut im Videospeich
er. Positive Werte für
[CNA] 120: ** Länge schreiben das Attribut nach rechts
, negative nach links.
[170] 121: */
[Z76] 122: void set_color(char color, int laenge)
[P70] 123: {
[X32] 124:     int i, ofs, s;
[000] 125:
[A12] 126:     regs.h.ah = 3;
[512] 127:     regs.h.bh = 0;
[T99] 128:     int86(0x10, &regs, &regs);
/* Cursorposition lesen */
[Y59] 129:     s = (laenge & 0x8000) ? -1 : 1;
/* Vorzeichen der Länge */

```

```

[PS8] 130:     laenge = (laenge < 0) ? -laenge : laeng
e; /* Absolutwert */
[F39] 131:     ofs = regs.h.dh * 160 + regs.h.dl * 2 +
s; /* Offset im Video- */
[UP3] 132:     /* speicher berechnen */
[RZ5] 133:     for (i = 0; i < laenge; ++i, ofs += (s
* 2))
[124] 134:         POKE(ofs, screen, color);
[U70] 135: }
[000] 136:
[1F6] 137: /*=====
=
[TC4] 138: ** Checksumme neu berechnen
[170] 139: */
[K93] 140: void checksum(void)
[P70] 141: {
[DB2] 142:     int chksum, i;
[FV8] 143:     chksum = 0; for (i = 0x10; i <= 0x2d; i
++) chksum += cmos[i];
[YEA] 144:     cmos[0x2e] = (char)(chksum & 0xff00) >
> 8; /* höherwertiges Byte */
[U2A] 145:     cmos[0x2f] = (char)chksum & 0x00ff;
/* niederwertiges Byte */
[U70] 146: }
[000] 147:
[1F6] 148: /*=====
=
[YR4] 149: ** Berechnung des Wochentages
[170] 150: */
[923] 151: char wochentag(DATE *dt)
[P70] 152: {
[0B3] 153:     long int jahr, monat;
[000] 154:
[ZY5] 155:     jahr = (dt->mon > 2) ? dt->jahr : dt->jahr
- 1;
[9C7] 156:     monat = (dt->mon > 2) ? dt->mon + 1 : d
t->mon + 13;
[F1A] 157:     return((char)((1461L * jahr / 4L + 153L
* monat / 5L + dt->tag - 2L) % 7L));
[U70] 158: }
[000] 159:
[1F6] 160: /*=====
=
[L35] 161: ** Byte in BCD-Format umwandeln.
[170] 162: */
[9F2] 163: char bcd(char n)
[P70] 164: {
[654] 165:     return(((n / 10) << 4) + n % 10);
[U70] 166: }
[000] 167:
[1F6] 168: /*=====
=
[FEC] 169: ** Datum und Zeit über MS-DOS-Funktionen 2A
hex bis 2Dhex lesen und schreiben.
[HYB] 170: ** Bei MS-DOS vor der Version 3.3 wird der
CMOS-Speicher nicht von den MS-DOS-
[PH4] 171: ** Funktionen aktualisiert.
[M5D] 172: ** Anstelle der C-Routinen »_dos_settime« u
nd »_dos_setdate« werden hier
[F4C] 173: ** die Systemuhr und der CMOS-Speicher glei
chzeitig direkt aktualisiert.
[K60] 174: **
[RE6] 175: /*=====
=
[962] 176: ** Datum holen
[170] 177: */
[WR3] 178: void get_date(DATE *dt)
[P70] 179: {
[UF2] 180:     regs.h.ah = 0x2a;
[TD3] 181:     intdos(&regs, &regs);
[Y43] 182:     dt->tag = regs.h.dl;
[663] 183:     dt->mon = regs.h.dh;
[L23] 184:     dt->jahr = regs.x.cx;
[PP3] 185:     dt->wt = wochentag(dt);
[U70] 186: }
[000] 187:
[1F6] 188: /*=====
=
[F02] 189: ** Zeit holen
[170] 190: */
[BX3] 191: void get_time(WORD *zt)
[P70] 192: {
[AU6] 193:     read_cmos(UHR); /* Array aktualisie
ren */
[WF2] 194:     regs.h.ah = 0x2c;
[V93] 195:     intdos(&regs, &regs);
[273] 196:     zt->std = regs.h.ch;
[D73] 197:     zt->min = regs.h.cl;
[M63] 198:     zt->sek = regs.h.dh;
[U70] 199: }
[000] 200:
[1F6] 201: /*=====
=
[FW2] 202: ** Datum schreiben
[170] 203: */
[5G5] 204: char set_date(WORD *neu, WORD *alt)
[P70] 205: {
[HYA] 206:     if (neu->tag == alt->tag && neu->mon ==
alt->mon && neu->jahr == alt->jahr)

```


CONTEXT PC

Autor: Matthias Uphoff



So logisch und einfach kann Textverarbeitung sein!

Was Context PC nicht kann:

- Dokumente mit 4000 Seiten verwalten
- 3-fach geschachtelte Fußnoten
- Grafik aus Fremdprogrammen einbinden.

Wenn Sie diese Features unbedingt benötigen, empfehlen wir Ihnen die Textverarbeitung der Firma XXX!

Was CONTEXT PC dafür jedoch ausgezeichnet beherrscht:

- einfachste Bedienerlogik, Hilfsmenues für jede denkbare Situation
- exakte Wiedergabe des Druckbildes auf dem Bildschirm
- wahlweise eigene oder WordStar-kompatible Tastaturbelegung
- Textbausteine auf Tastendruck in Ihren Text einfügen
- Texte bis 65 Zeilen einfach mit einer Taste belegen
- perfekte Silbentrennung
- Rechnen im Text, automatische Datum- und Zeiteinfügung, Kalender
- Grafik-Zeichen direkt aus Tabelle in Text einsetzen
- Es können gleichzeitig 5 Texte unabhängig bearbeitet werden.
- Formatierung, die keine Wünsche offenläßt
- zweispaltiges Formatieren
- automatische BackUp-Funktion
- integrierter Dateimanager
- DOS-Aufrufe (auch andere Programme) aus ConText heraus starten
- durch hochspezialisierte Bildschirmtreiber auch im Graphikbetrieb außerordentlich schnell
- vielfältiger Einsatz in Batch-Dateien durch zahlreiche Möglichkeiten der Parameterübergabe
- hervorragend als Programm-Editor
- Programm im Hauptspeicher, keinerlei Diskettenzugriffe bei der Bearbeitung, ideal für Computer mit einem Laufwerk
- minimaler Speicherplatzbedarf bei maximaler Leistung
- durch äußerst effiziente Programmierung durch Assembler
- umschaltbare Textdarstellung während der Bearbeitung: 25/35 Zeilen (EGA + Hercules) 20/25 Zeilen (CGA)
- alle Standards einer Textverarbeitung: Suchen/Ersetzen auch mit Wild-Cards, Textmarker, jederzeit änderbare Rand-einstellung, Blocksatz, Trennhilfe, 240 Zeichen Zeilenbreite, automatischer Tabulator, Zeile zentrieren
- arbeitet mit allen handelsüblichen Druckern
- umfangreiches Handbuch ca. 190 Seiten mit ausführlichen Erklärungen und Beispielen

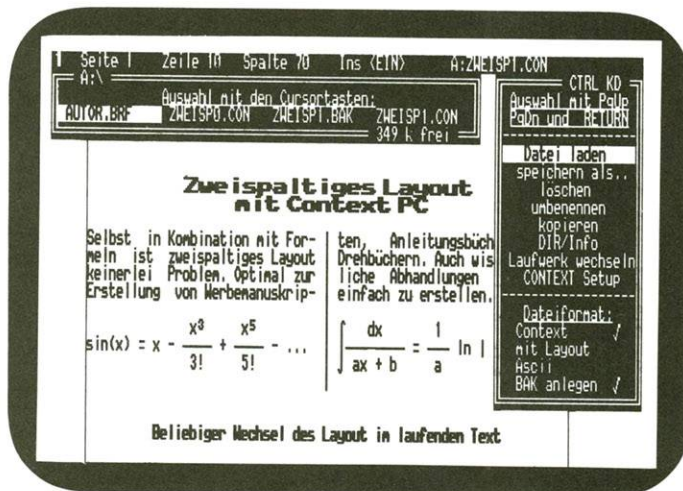
Notwendige Konfiguration:

PC XT/AT mit 256 kByte oder mehr, 1 Diskettenlaufwerk bzw. Festplatte, MS-DOS ab Version 2.0. CGA-, Hercules- und EGA-Grafikkarten werden unterstützt.

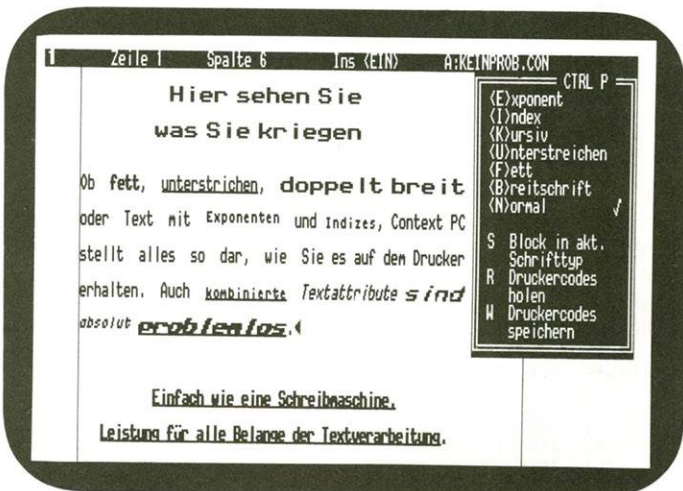
Zum Lieferumfang gehört ein umfangreiches deutsches Handbuch inklusive Tutorial.

© 1988 DMV GmbH

Best.-Nr. 228 5 1/4" Disk. Best.-Nr. 234 3 1/2" Disk.



Der Dateimanager des Context PC macht DOS-Benutzeroberflächen überflüssig.



»WYSIWYG« in Reinform: Alle Schriftattribute werden am Bildschirm dargestellt.

CONTEXT PC



99,- DM (unverbindliche Preisempfehlung)

Wenn Sie über den DMV-Bestellservice bestellen, gilt folgendes:

Inland:		Ausland:	
Einzelpreis	99,- DM	Einzelpreis	99,- DM
zzgl. Versandkosten	3,- DM	zzgl. Versandkosten	5,- DM
Endpreis	102,- DM	Endpreis	104,- DM

- Bitte Bestellkarte benutzen -

DMV GmbH · Abt. Software · Postfach 250 · 3440 Eschwege


```

[TL1] 207: return(0);
[VF2] 208: regs.h.ah = 0x2b;
[3A3] 209: regs.h.dl = neu->tag;
[5B3] 210: regs.h.dh = neu->mon;
[A73] 211: regs.x.cx = neu->ja;
[V93] 212: intdos(&regs, &regs);
[E12] 213: if (regs.h.al)
[WT1] 214: return(-1);
[3V5] 215: cmos[6] = neu->wt = wochentag(neu);
[2S7] 216: outp(ADDR, 6); /* Wochentag a
ktualisieren */
[FB8] 217: outp(DATA, cmos[6]); /* wird von DO
S > Version 3.0 */
[VU7] 218: cmos[7] = bcd(neu->tag); /* nicht unter
stützt */
[DV3] 219: cmos[8] = bcd(neu->mon);
[GB5] 220: cmos[9] = bcd((char)(neu->ja % 100));
[TW5] 221: cmos[0x32] = bcd((char)(neu->ja / 100));
;
[TL1] 222: return(0);
[U70] 223: }
[000] 224:
[1F6] 225: /*=====
=
[GN2] 226: ** Zeit schreiben
[170] 227: */
[HP5] 228: char set_time(TIME *neu, TIME *alt)
[P70] 229: {
[4HB] 230: if (neu->std == alt->std && neu->min ==
alt->min && neu->sek == alt->sek)
return(0);
[TL1] 231: return(0);
[XF2] 232: regs.h.ah = 0x2d;
[0C3] 233: regs.h.ch = neu->std;
[MA3] 234: regs.h.cl = neu->min;
[CA3] 235: regs.h.dh = neu->sek;
[L87] 236: regs.h.dl = 0; /* Hundertstel
Sekunden */
[XK3] 237: intdos(&regs, &regs);
[E12] 238: if (regs.h.al)
[WT1] 239: return(-1);
[PZ3] 240: cmos[0] = bcd(regs.h.dh);
[904] 241: cmos[2] = bcd(regs.h.cl);
[Z23] 242: cmos[4] = bcd(regs.h.ch);
[TL1] 243: return(0);
[U70] 244: }
[000] 245:
[1F6] 246: /*=====
=
[ZYB] 247: ** Datum und Uhrzeit im CMOS-Speicher aktua
lisieren (MS-DOS-Emulation)
[170] 248: */
[3H3] 249: void write_kal(void)
[P70] 250: {
[3G1] 251: int i = 0;
[EN8] 253: while(i < 9) { /* Datum/Zeit
ins CMOS schreiben */
[S42] 254: outp(ADDR, i);
[7E3] 255: outp(DATA, cmos[i++]);
[U70] 256: }
[SE8] 257: outp(ADDR, 0x32); /* Jahrhundert
-Byte schreiben */
[FT3] 258: outp(DATA, cmos[0x32]);
[U70] 259: }
[1F6] 260: /*=====
=
[4SA] 261: ** Eingabe Routine - wartet auf Befehl oder
Byte Nummer (00 - 3Fhex)
[170] 262: */
[6P2] 263: int input(void)
[P70] 264: {
[A22] 265: char st[81];
[549] 266: static char gueltig[] = "HTLSWRDQ"; /*
gültige Befehle */
[CP0] 267: int
[AR0] 268: i = 0,
[MT7] 269: len = 0, /* L
änge des Eingabe-String */
[MX5] 270: cmd = -1; /* R
ückgabewert */
[000] 271:
[933] 272: while ((len && i < 80) {
[951] 273: cln(24);
[LH5] 274: printf(" Befehl oder Byte Nr. ? > "
);
[MK7] 275: set_color(icolor, 80 - gety()); /* H
ervorhebung */
[UG4] 276: set_color(icolor, - gety());
[000] 277:
[MD9] 278: if (!getstr(st)) return(ESC); /* S
tring-Eingabe abwarten */
[S87] 279: strupr(st); /* g
roße Buchstaben */
[000] 280:
[H04] 281: if ((len = strlen(st)) == 1)
[R37] 282: && (strcspn(gueltig
, st)) < strlen(gueltig))
[KB2] 283: return(st[0]);

```

```

[000] 284:
[VH6] 285: else if ((len > 2) || !sscanf(st,
"%2x", &cmd))
[M41] 286: len = 0;
[XS3] 287: if (cmd < 0 || cmd > 0x3f)
[M41] 288: len = 0;
[U70] 289: }
[EY1] 290: return(cmd);
[U70] 291: }
[000] 292:
[1F6] 293: /*=====
=
[Y55] 294: ** Einfacher Texteingabe-Editor
[U75] 295: ** - Abbruch mit Escape-Taste
[847] 296: ** - Löschfunktion mit Backspace-
Taste
[170] 297: */
[EF3] 298: int getstr(char *st)
[P70] 299: {
[072] 300: char ch, *i = 0;
[000] 301:
[M41] 302: i = st;
[CR7] 303: while ((ch = (char)toupper((char)getch(
))) != '\r') {
[122] 304: switch (ch) {
[0G6] 305: case BACKSP : { /* Backsp
ace-Taste */
[KV1] 306: if (i > st) {
[EZ1] 307: write_ch(8);
[732] 308: write_ch(32);
[EZ1] 309: write_ch(8);
[0K0] 310: --i;
[U70] 311: }
[P01] 312: break;
[U70] 313: }
[227] 314: case NULL: { /* Extend
ed Keys löschen */
[R61] 315: getch();
[P01] 316: break;
[U70] 317: }
[RB5] 318: case ESC : { /* Escape
Taste */
[YB2] 319: return(*st = 0);
[UY6] 320: } /* Zeiche
n in String speichern */
[E32] 321: default : {
[181] 322: *i++ = ch;
[F72] 323: write_ch(ch);
[U70] 324: }
[U70] 325: }
[U70] 326: }
[8X0] 327: *i = 0;
[RS1] 328: return (1);
[U70] 329: }
[000] 330:
[7C6] 331: /*=====
=
[000] 332: /

```

Listing 4. In »cmosutil.c« befinden sich die BIOS-Zugriffe

```

[L13] 1: #include <stdio.h>
[9L2] 2: #include <dos.h>
[203] 3: #include <conio.h>
[000] 4:
[UW0] 5: main()
[P70] 6: {
[WE1] 7: perror();
[7Z0] 8: inp();
[M71] 9: outp();
[CY2] 10: _dos_findfirst();
[SR2] 11: _dos_findnext();
[U70] 12: }

```

Listing 5. Die Funktionen von qcsbcs.c« müssen Sie in das Quick-C-Library integrieren

Snapshot: Bildschirmfotos per Software

Zuweilen steht der PC-Anwender vor der Notwendigkeit, seinen Text- oder Grafikbildschirm auf Diskette abzuspeichern; zum Beispiel wenn in Ermangelung eines eigenen Druckers Hardcopies auf einem anderen Gerät zu Papier gebracht werden müssen, erspart dies eine Menge Arbeit.

Snapshot (Listing 1) ist ein residentes Programm, das sich nach Aufruf sofort in den Hintergrund zurückzieht. Dort wartet es darauf, daß es vom Anwender mit [Alt-PrtSc] aufgerufen wird, um seine eigentliche Aufgabe zu erfüllen. Diese besteht darin, den momentan sichtbaren Bildschirm in komprimierter Form als ausführbare Exe-Datei auf Diskette oder Festplatte zu speichern, und sich dann wieder »schlafenzulegen«.

Das hört sich einfach an, ist aber angesichts der vielen Grafikstandards (MDA, CGA, EGA, MCGA und VGA) ein wahres Verwirrspiel. Hinzu kommt, daß einige Computer (Schneider, Olivetti) und fast alle Karten-Nachbauer die vorgegebenen Möglichkeiten des Originals übertreffen wollten und zusätzliche Grafikmodi einführten, die natürlich zu nichts außer zu sich selbst kompatibel sind. Die bekannte Hercules-Karte ist zum Beispiel ein solcher Zeitgenosse. Auch wenn sie sich inzwischen zum Pseudo-Standard entwickelt hat, so ist ihre Herkunft doch nicht zu übersehen: Der Grafikmodus wird in keinstrengster Weise vom BIOS unterstützt. Das hat zur Folge, daß man auf umständliches »Registergefummel« zurückgreifen muß, wenn man Grafik ausgeben möchte.

Diese Tatsachen haben dazu geführt, daß bei der Entwicklung des Programms viel Wert auf die Erweiterbarkeit gelegt wurde. Sie können es somit auch mit Ihrer speziellen Grafikkarte verwenden. Die Sondermodi Hercules und Schneider 2 sind allerdings bereits integriert, da sie ziemlich verbreitet sind. Und sollten Sie eine EVA/1024-VGA-Karte besitzen, so brauchen Sie sich um deren spezielle Modi ebenfalls nicht mehr zu kümmern.

Die verwirrende Vielfalt bei den Grafikstandards beschert Probleme

Da einige Programme (zum Beispiel MS-Windows) mit eigenem Tastaturreiber arbeiten, wurde eine Routine in den Ticker-Interrupt eingehängt, die den Tastaturvektor ständig überwacht. Erfährt nun dieser Vektor vom Anwenderprogramm eine Veränderung, so würde die Tastenkombination [Alt-PrtSc] nicht mehr erkannt. In diesem Fall »drängelt« sich Snapshot einfach wieder an die erste Position vor.

Leider haben einige Programme (AFD und Quick Basic) für solch ein Vorgehen kein Verständnis und »hängen sich auf«. Allerdings kann auch der Interrupt 05hex, den das Betätigen von [Shift-PrtSc] aufruft, das Programm aktivieren. Geben Sie hierzu der Konstante »intnr« den Wert 5. Wenn Sie keinen MASM ab Version 4.0 besitzen, dann müssen Sie den Hexdump (Listing 2) mit dem Programm »SCheck« eingeben. In diesem Fall bleibt Ihnen jedoch die variable Interrupt-Anpassung verwehrt, Sie müssen sich mit den vorgegebenen Video-Modi begnügen. Andernfalls erzeugen Sie aus der Quelldatei mit folgenden drei Befehlen die ausführbare Com-Datei:

```
masm snapshot;
link snapshot;
exe2bin snapshot snapshot.com
```

Rufen Sie dann das Programm mit folgenden Parametern auf:

```
snapshot [Dateiname]
[/Option] [/Option] [...]
```

Der Dateiname kann Angaben über Laufwerk und Pfad enthalten und darf maximal 62 Zeichen umfassen. Unter diesem Namen wird der Bildschirm gespeichert, wenn Sie [Alt-PrtSc] betätigen. Falls es die entsprechende Datei schon gab, wird sie ohne Rückfrage gelöscht. Damit es nicht erforderlich ist, bei jedem Bild einen neuen Namen einzugeben, werden doch Ziffern, die sich im Namen oder in der Erweiterung befinden, bei jedem erfolgreichen Aufruf weitergezählt. Aus »C:\BILD000.EXE« wird also »C:\BILD001.EXE« und so weiter. Lassen Sie den Namen weg, so wird der vorangehende übernommen; beim Erstaufruf ist dies der Name unter dem Label »pfad«.

Eigenbrötlerische Spiele meiden aus Geschwindigkeitsgründen das BIOS

Leider halten sich nicht alle Programme an die Modusumschaltung per BIOS. Um in einem derartigen Fall nicht »Müll« abzuspeichern, kann man einen bestimmten Modus per Schalter erzwingen. Die Syntax dafür ist »/Modus[,Seite]«. »<Modus>« steht für eine zweistellige Hexadezimalzahl von 00hex bis 7Fhex; die Bedeutung dieser Zahl entnehmen Sie bitte den Handbüchern zu Computer und Videoadapter. Die Angabe der Seite kann entfallen, es wird dann der vorherige Wert übernommen. Die Numerierung der Seiten beginnt übrigens bei »0«. Ein Beispiel soll die Arbeit mit diesem Schalter verdeutlichen. Sie besitzen eine EGA-Karte und wollen die Titelgrafik des Adventures »The Pawn« abspeichern. Der erste Versuch scheitert daran, daß zwar ein sinnvolles Bild dargestellt wird, dieses aber keine Grafik, sondern nur den Schriftzug »Pondering...« enthält. Es liegt die Vermutung nahe, daß das Adventure am BIOS vorbei auf die zweite Seite umgeschaltet hat. Aus der Auflösung des Bildes läßt sich mit etwas Übung der verwendete Video-Modus erkennen. Hier sind es 640 x 350 Bildpunkte in 16 Farben, was dem EGA-Modus 10hex entspricht. Sie starten Snapshot erneut und geben als Option »/10,1« an – und diesmal klappt es auch. Wollen Sie, daß die Erkennung des Video-Modus wieder automatisch erfolgt, so rufen Sie Snapshot mit dem Schalter »/a« auf.

Für Besitzer mancher Videokarten, die unter anderem auch eine Hercules-Emulation bieten, wurden ebenfalls zwei Schalter vorgesehen. Bei der Hercules-Karte läßt sich nämlich der aktuelle Status (Text oder Grafik) nur durch Vergleichen des Lightpen-Registers mit einem bestimmten Schwellwert auslesen. Manche »kompatible« Karten verzichten allerdings auf den Lightpen-Anschluß, so daß eine Unterscheidung hier nicht möglich ist. Aber auch die Originalkarte besitzt einen Nachteil: Man kann die eingestellte Seite nicht bestimmen. Ordentliche Programme setzen zwar die entsprechenden Bits 1 und 7 in der Speicherzelle 0040:0065hex, aber darauf sollte man sich nicht unbedingt verlassen. Man kann also auch hier wieder nur mit einem Schalter für Klarheit sorgen. Dieser heißt »/h[,Seite]«. Ist er aktiviert, so wird immer, wenn der Video-Modus 07hex eingeschaltet ist, die Hercules-Grafik der entsprechenden Seite abgespeichert. Danach schaltet »/m« (wie MDA) wieder auf automatische Hercules-Erkennung um.

Hin und wieder kann es vorkommen, daß nach dem Betätigen von [Alt-PrtSc] ein Ton ausgegeben wird.

Dies entspricht einer Fehlermeldung, die auf vier Ursachen hinweisen kann:

- der momentan eingestellte Video-Modus ist unbekannt,
- der angegebene Pfad ist fehlerhaft,
- es trat ein Schreibfehler auf (wird erst ab MS-DOS 3.0 abgefangen)
- auf der Diskette ist nicht mehr genügend Platz vorhanden.

In einigen der genannten Fälle wird zwar eine Exe-Datei erzeugt, ihre Ausführung jedoch durch das Betriebssystem verhindert.

Wollen Sie wissen, welche Videomodi Snapshot verarbeiten kann, so sehen Sie sich das Assemblerlisting an. Der erste Block gilt für die von IBM unterstützten Modi CGA, MDA, EGA, MCGA und VGA, wobei den Nummern 06 und 07 eine besondere Bedeutung zukommt. Sie stehen nämlich für jeweils zwei Modi: CGA und Schneider 2, beziehungsweise MDA und Hercules. Im einzelnen muß in den dazugehörigen Unterprogrammen geklärt werden, welcher Modus denn nun wirklich aktiv ist. Von Interesse ist in diesem Zusammenhang die Erkennung des Schneidermodus mit 640 x 200 in 16 Farben. So seltsam es auch klingen mag: Der Schneider PC kennt überhaupt keinen CGA-Modus 06! Er befindet sich in diesem Fall immer im Grafikmodus 2, nur haben hierbei alle vier Bildebenen denselben Inhalt. Man muß also die Grafikebenen Byte für Byte vergleichen, um unterscheiden zu können.

Der zweite Block bei den Videomodi ist speziell für die EVA/1024 gedacht. Besitzen Sie eine andere Karte, dann löschen Sie ihn nebst dazugehöriger Unterprogrammen. Sie können auch die Unterprogramme für den Schneider- und den Hercules-Modus weglassen, müssen dann allerdings die Labels bei den Videomodi in »cga« ändern. Falls Sie die EGA- beziehungsweise VGA-Modi nicht benötigen, ist es zweckmäßig, zusätzlich zu den Unterprogrammen auch bei der Modusdefinition alle Zeilen mit einem entsprechenden Label am Schluß zu löschen.

Snapshot ist für Erweiterungen auf andere Grafikkarten und deren Modi jederzeit offen

Wenn Sie neue Videomodi hinzufügen wollen, dann sollten Sie den Aufbau einer Definitionszeile kennen. Eingefügt wird diese vor dem Label »videoende«, sie hat folgenden Inhalt: »v <Modus, Segment, Seitenfaktor, Länge, Wortzugriff, Unterprogramm>«. Die Zahl bei »Modus« entspricht derjenigen beim Aufruf der BIOS-Funktionen 00hex und 0Fhex. Das »Segment« steht für den Anfang der ersten Bildschirmseite. Der »Seitenfaktor« wird mit 256 multipliziert und gibt dann den Abstand von einer Bildschirmseite zur nächsten in Byte an. Die »Länge« wird ebenfalls in Byte angegeben, wobei der Wert 0 für 65536 steht. »Wortzugriff« ist ein Flag, das vor allem für die Effizienz des Packers verantwortlich ist. In den Textmodi hätte das byteweise Suchen nach sich wiederholenden Daten nämlich wenig Sinn, da zwischen zwei Zeichen immer ein Attribut-Byte liegt. Das »Unterprogramm« schließlich ist für das eigentliche Abspeichern verantwortlich. An dieser Stelle können auch Unterscheidungen zwischen ansonsten gleichen Modi eingebaut werden (siehe Schneider und Hercules). Die Namen der Unterprogramme lassen nicht unbedingt auf die verwendete Kartenemulation schließen, so wird zum Beispiel »cga« auch von dem »ein-ebenen« VGA-Modus aufgerufen.

Im Prinzip brauchen Sie für die Entwicklung eigener Unterprogramme nur drei Routinen und deren Parameter zu kennen:

Die Routine »vorspann« wird nur einmal am Anfang aufgerufen. Sie zerstört AX, DX und SI und benötigt in »neuermodus« den gewünschten Modus, in »videobasis« das Bildschirmsegment und in »wort« das Flag für den Wort-

zugriff. Diese drei Werte werden bereits vom Hauptprogramm aus der Definitionszeile gelesen und eingesetzt. Die Routine »schreiben« schreibt einen Datenblock, der durch DS:SI adressiert ist, auf Diskette. Das erste Wort dieses Blocks muß dessen Länge enthalten. AX und SI werden zerstört.

Am wichtigsten ist die Routine »packen«. Sie schreibt den Block, der durch videobasis:BX adressiert ist und eine Länge von CX Byte besitzt, nebst einem Aufruf des Entpackers auf Diskette. Das Hauptprogramm berechnet BX je nach aktueller Seite, CX übernimmt es direkt aus der Definitionszeile. Der Packer zerstört AX und SI.

DS, ES und DI dürfen nicht verändert werden, da sie für den Variablenzugriff und das Schreiben auf Diskette zuständig sind. Ihre eigentliche Aufgabe besteht nun darin, zwischen diesen Unterprogrammaufrufen die einzelnen Ebenen des Grafikadapters zum Lesen umzuschalten und die entsprechenden Umschaltbefehle für das spätere Entpacken auf Diskette zu schreiben. Sind Sie damit fertig, dann »springen« Sie mit »ret« zum Hauptprogramm zurück. Hat Ihr Grafikadapter allerdings nur eine Ebene, dann genügt eine Definition bei den Videomodi mit »cga« als Label.

Eine künstlerische Betätigung: Farben nachträglich verändern

So umfangreich und komfortabel das BIOS der VGA-Karten auch sein mag, es stellt den Anwender doch vor schier unlösbare Probleme, wenn er bei angeschlossenem Schwarzweiß-Monitor die Farbpalette auslesen möchte. Wenn man zum Beispiel die Werte 0/0/0 setzt, dann bekommt man mit den entsprechenden Funktionen 1010hex und 1012hex die Folge 7/7/7 zurück. Daran ändert auch das Abschalten des »Gray Scale Summing« nichts. Einige Spiele setzen im EGA-Betrieb die Farben nicht mit den BIOS-Routinen, die von Snapshot abgefangen werden, so daß die Werte unwiederruflich in den Tiefen des Video-Controllers verlorengehen. Für derartige Fälle ist das Programm »palette.pas« (Listing 3) zuständig. Es erlaubt die Einstellung der einzelnen Farben bei den CGA-Multicolormodi 04hex und 05hex, sowie allen EGA-/VGA-Grafikmodi außer Nummer 0Fhex (EGAMono). Compilieren Sie »palette.pas« mit Turbo Pascal 4.0 auf Diskette und rufen Sie es mit

```
palette <bildname>
```

auf. »bildname« ist eine Datei, die vorher mit Snapshot erzeugt wurde, wobei die Erweiterung ».exe« fehlen kann. Die Bedienung des Programms ist vom Video-Modus abhängig. Bei CGA-Bildern können Sie mit [p] die Palette und mit [h] die Hintergrundfarbe auswählen. Anschließend speichern Sie das Bild entweder mit [Return], oder Sie brechen das Programm mit [Esc] ab. Bei EGA- und VGA-Bildern ist angesichts der Farbenvielfalt eine solche einfache Bedienung nicht mehr möglich. Hier müssen Sie zuerst mit den Cursor-Tasten [Links], [Rechts], [Home], [End] sowie den Tastenkombinationen [Ctrl-Links] und [Ctrl-Rechts] eine Farbe anwählen. Zur Orientierung blinkt die entsprechende Farbe dabei kurz auf. Wenn Ihnen die momentane Position dieses »Cursors« entfallen ist, dann können Sie das Blinken mit der Leertaste wiederholen. Die aktuelle Farbe verändern Sie nun mit den Anfangsbuchstaben der englischen Grund- und Mischfarben, also »red«, »green«, »blue«, »cyan«, »yellow«, »magenta« und »white«. Diese Tasten geben normalerweise den betreffenden Farbanteil dazu; drückt man sie jedoch zusammen mit [Shift], so wird die Farbe weggenommen. Wollen Sie eine Farbe völlig neu einstellen, so setzen Sie sie vorher mit [0] auf Schwarz. Zum Schluß sichern Sie die mühsam erarbeitete Palette mit [Return] auf Diskette. Übrigens, mit Snapshot erzeugte Bildprogramme erzeugen das Bild und warten dann auf einen Tastendruck.

Wollen Sie in eigenen Demonstrationsprogrammen Snapshots darstellen, aber den Benutzer nicht zu wiederholtem Tastendruck nötigen, rufen sie die Bild-Datei mit der Option »/c« auf, das Programm wartet dann nicht. Natürlich schaltet es den Video-Modus auch nicht mehr zurück, der Betrachtungsgenuß eines Bildes, das nur eine halbe Sekunde auf dem Schirm zu sehen ist, hielte sich wahrscheinlich in Grenzen. Ihr Programm hätte dann selbst dafür zu sorgen, daß nachfolgende Darstellungen anderer Art lesbar werden, zum Beispiel mittels der Darstellung eines Snapshot in der dann gewünschten Bildschirmauflösung.

(Wolfgang Lorenz/db)

```
[PC3] 1: code segment public
[8K3] 2: assume cs:code,ds:code
[3A1] 3: org 0100h
[000] 4:
[7V5] 5: ;Interrupt 5 (PrtSc) oder 9 (KbdInt)
[4D1] 6: intrnr = 9
[000] 7:
[201] 8: begin:
[5X2] 9: jmp installieren
[000] 10:
[2M7] 11: ;Struktur für jeweils eine Definitionsz
    eile
[VA1] 12: v struc
[1K1] 13:   modus db ?
[6K1] 14:   basis dw ?
[7S1] 15:   faktor db ?
[8R1] 16:   laenge dw ?
[XE1] 17:   text db ?
[N12] 18:   routine dw ?
[FZ0] 19: v ends
[H2C] 20: ;Länge der Struktur in Bytes, die MASM-
    Funktion "length" mag hier nicht
[SL2] 21: lengthvideo = 9
[000] 22:
[42B] 23: ;Definition der von IBM unterstützten V
    ideomodi außer PCjr
[1T1] 24: videomodi:
[8M4] 25: v <00h,0b800h,8,2000,1,cgab> ;40*25
[AM4] 26: v <01h,0b800h,8,2000,1,cgab> ;40*25
[6N4] 27: v <02h,0b800h,16,4000,1,cgab> ;80*25
[8N4] 28: v <03h,0b800h,16,4000,1,cgab> ;80*25
[985] 29: v <04h,0b800h,64,16192,0,cgac> ;320*200*4
[B85] 30: v <05h,0b800h,64,16192,0,cgac> ;320*200*4
[MA6] 31: v <06h,0b800h,64,16192,0,schneider> ;640*200
    *2
[PE5] 32: v <07h,0b000h,16,4000,1,hercules> ;80*25
[Y35] 33: v <0dh,0a000h,32,8000,0,ega> ;320*200*16
[W65] 34: v <0eh,0a000h,64,16000,0,ega> ;640*200*16
[B65] 35: v <0fh,0a000h,128,28000,0,ega> ;640*350*4
[W95] 36: v <10h,0a000h,128,28000,0,ega> ;640*350*16
[D25] 37: v <11h,0a000h,0,38400,0,cga> ;640*480*2
[545] 38: v <12h,0a000h,0,38400,0,ega> ;640*480*16
[475] 39: v <13h,0a000h,0,64000,0,vga> ;320*200*256
[000] 40:
[Z56] 41: ;Definition der Videomodi der EVA/1024
[UR4] 42: v <18h,0b000h,64,11616,1,cga> ;132*44
[3N4] 43: v <19h,0b000h,16,6600,1,cga> ;132*25
[TN4] 44: v <1ah,0b000h,16,7392,1,cga> ;132*27
[754] 45: v <22h,0b800h,64,11616,1,cga> ;132*44
[MN4] 46: v <23h,0b800h,32,6600,1,cga> ;132*25
[WN4] 47: v <24h,0b800h,32,7392,1,cga> ;132*27
[C45] 48: v <25h,0a000h,0,38400,0,ega> ;640*480*16
[1P4] 49: v <26h,0b800h,64,9600,1,cga> ;80*60
[C25] 50: v <29h,0a000h,0,60000,0,ega> ;800*600*4
[UN4] 51: v <2ah,0b800h,32,8000,1,cga> ;100*40
[HT4] 52: v <2dh,0a000h,0,0,0,eva> ;640*350*256
[UT4] 53: v <2eh,0a000h,0,0,0,eva> ;640*480*256
[5R4] 54: v <37h,0a000h,0,0,0,ega> ;1024*768*16
[000] 55:
[4F6] 56: ;Hier können Sie eigene Modi einfügen
[000] 57:
[B02] 58: videoende = $
[000] 59:
[SV5] 60: ;Blinken ein/aus für EGA-Textmodi
[NS0] 61: cgab:
[T82] 62: call vorspann
[KB3] 63: mov si,offset blink
```

```
[ZC2] 64: call schreiben
[9S1] 65: jmp packen
[000] 66:
[7N3] 67: blink dw blinke-blinka
[V81] 68: blinka:
[EV1] 69: mov ax,1003h
[8A1] 70: mov bl,1
[8A3] 71: blinken = byte ptr $-1
[Y71] 72: int 10h
[8D1] 73: blinke = $
[000] 74:
[GR7] 75: ;Einfache Modi ohne Farbpaletten und Bi
    tplanes
[TM0] 76: cga:
[T82] 77: call vorspann
[9S1] 78: jmp packen
[000] 79:
[S24] 80: ;CGA-Modi 320*200 in 4 Farben
[PS0] 81: cgac:
[T82] 82: call vorspann
[6P3] 83: mov si,offset cgafarbe
[ZC2] 84: call schreiben
[9S1] 85: jmp packen
[000] 86:
[FK3] 87: cgafarbe dw cgafe-cgaf
[TS0] 88: cgaf:
[3V2] 89: ;Randfarbe setzen
[NM1] 90: mov ah,0bh
[SE1] 91: mov bx,16
[183] 92: cgarand = byte ptr $-2
[Y71] 93: int 10h
[DT2] 94: ;Palette setzen
[NM1] 95: mov ah,0bh
[TL1] 96: mov bx,256
[KS3] 97: cgapalette = byte ptr $-2
[Y71] 98: int 10h
[B71] 99: cgafe = $
[000] 100:
[1B7] 101: ;Prüfen ob Schneidernodus 2 oder CGA-Mo
    dus 06
[9R1] 102: schneider:
[TC1] 103: push bx
[LH1] 104: xor bx,bx
[XL1] 105: mov ah,06h
[D81] 106: int 15h
[GWC] 107: ;Bei Schneider-Computern enthält BX nac
    h diesem Interrupt die BIOS-Version
[AE1] 108: and bx,bx
[T31] 109: pop bx
[RV0] 110: je cga
[37A] 111: ;Die vier Bitplanes des Schneider PC au
    f Gleichheit prüfen
[TC1] 112: push bx
[WC1] 113: push cx
[BC1] 114: push ds
[9Y2] 115: mov ds,videobasis
[2X3] 116: ;Port für Lesezugriffe
[XZ1] 117: mov dx,03deh
[VG1] 118: xor si,si
[SU1] 119: mov cx,8096
[X52] 120: pruefewort:
[1Y3] 121: ;Wert von Seite 0 lesen
[5A1] 122: mov al,0
[6H1] 123: out dx,al
[VH2] 124: mov bx,ds:[si]
[AY3] 125: ;Mit Seite 1 vergleichen
[E01] 126: inc al
[6H1] 127: out dx,al
[YH2] 128: cmp ds:[si],bx
[YW4] 129: ;Ungleich => Schneidernodus 2
[R73] 130: jne schneidermodus
[BY3] 131: ;Mit Seite 2 vergleichen
[E01] 132: inc al
[6H1] 133: out dx,al
[YH2] 134: cmp ds:[si],bx
[R73] 135: jne schneidermodus
[P71] 136: ;Seite 3
[E01] 137: inc al
[6H1] 138: out dx,al
[PW0] 139: lodsw
[TF1] 140: cmp ax,bx
[UW2] 141: loope pruefewort
[DT2] 142: schneidermodus:
[E31] 143: pop ds
[V31] 144: pop cx
[T31] 145: pop bx
[RV0] 146: je cga
[S56] 147: ;Bild im Schneidernodus 2 abspeichern
```

Listing 1. Der Quellcode des Programms »snapshot.asm«


```

[T82] 148: call vorspann
[B54] 149: mov si,offset schneider0
[ZC2] 150: call schreiben
[K62] 151: ;Lesen Seite 0
[X21] 152: mov dx,03deh
[5A1] 153: mov al,0
[6H1] 154: out dx,al
[TY1] 155: call packen
[C54] 156: mov si,offset schneider1
[ZC2] 157: call schreiben
[N62] 158: ;Lesen Seite 1
[6A1] 159: mov al,1
[6H1] 160: out dx,al
[TY1] 161: call packen
[D54] 162: mov si,offset schneider2
[ZC2] 163: call schreiben
[S62] 164: ;Lesen Seite 2
[7A1] 165: mov al,2
[6H1] 166: out dx,al
[TY1] 167: call packen
[E54] 168: mov si,offset schneider3
[ZC2] 169: call schreiben
[V62] 170: ;Lesen Seite 3
[8A1] 171: mov al,3
[6H1] 172: out dx,al
[9S1] 173: jmp packen
[000] 174:
[6F4] 175: schneider0 dw schneider1-s0
[YG0] 176: s0:
[0R5] 177: ;Alle 4 Bitplanes werden angezeigt
[DY1] 178: mov dx,03d9h
[9N1] 179: mov al,0fh
[6H1] 180: out dx,al
[DX2] 181: ;Schreiben Seite 0
[K71] 182: add dx,4
[6A1] 183: mov al,1
[6H1] 184: out dx,al
[EF4] 185: schneider1 dw schneider2-s1
[1H0] 186: s1:
[GX2] 187: ;Schreiben Seite 1
[7A1] 188: mov al,2
[6H1] 189: out dx,al
[PF4] 190: schneider2 dw schneider3-s2
[4H0] 191: s2:
[LX2] 192: ;Schreiben Seite 2
[9A1] 193: mov al,4
[6H1] 194: out dx,al
[FG4] 195: schneider3 dw schneidere-s3
[7H0] 196: s3:
[PX2] 197: ;Schreiben Seite 3
[DA1] 198: mov al,8
[6H1] 199: out dx,al
[L32] 200: schneidere = $
[000] 201:
[T47] 202: ;Unterscheidung zwischen Hercules und M
DA
[LK1] 203: hercules:
[A57] 204: ;Vom Benutzer erzwungener Herculesmodus
[H54] 205: cmp byte ptr hercmodus,0
[WC1] 206: jne herc
[654] 207: mov byte ptr hercseite,0
[G97] 208: ;Bit 1 und 7 der BIOS-Variable 40:65h a
uswerten
[BC1] 209: push ds
[5N1] 210: mov ax,40h
[1H1] 211: mov ds,ax
[5S2] 212: mov al,ds:[65h]
[E31] 213: pop ds
[1S3] 214: ;Bit 1: 0=Text, 1=Grafik
[UE1] 215: test al,2
[Y21] 216: je hercules?
[S44] 217: ;Bit 7: 0=Seite 0, 1=Seite 1
[EU1] 218: test al,128
[R51] 219: je herc
[754] 220: mov byte ptr hercseite,1
[1E2] 221: jmp short herc
[DS6] 222: ;Herculeskarte auf Hardwarebasis suchen
[BS1] 223: hercules?:
[LB4] 224: ;Lightpen-Flipflop setzen
[7Y1] 225: mov dx,03b9h
[6H1] 226: out dx,al
[E35] 227: ;Bit 2 des Status-Port gesetzt?
[S11] 228: inc dx
[161] 229: in al,dx
[UE1] 230: test al,2
[WY5] 231: ;Nein, dann kein Lightpen vorhanden
[VV0] 232: je mda

```

```

[GM4] 233: ;Lightpen-Flipflop löschen
[S11] 234: inc dx
[6H1] 235: out dx,al
[3V2] 236: ;Bit 2 gelöscht?
[L01] 237: dec dx
[161] 238: in al,dx
[UE1] 239: test al,2
[H36] 240: ;Ja, dann Text-/Grafikmodus prüfen
[7D1] 241: je warte0
[4N0] 242: mda:
[U91] 243: jmp cga
[NM5] 244: ;Warten auf Rasterstrahl im Bild
[F81] 245: warte0:
[161] 246: in al,dx
[EU1] 247: test al,128
[7D1] 248: je warte0
[2U5] 249: ;Warten auf Rücklauf des Strahls
[K81] 250: wartel:
[161] 251: in al,dx
[EU1] 252: test al,128
[WP1] 253: jne wartel
[FD3] 254: ;Flipflop set & reset
[S11] 255: inc dx
[6H1] 256: out dx,al
[Z91] 257: sub dx,2
[6H1] 258: out dx,al
[AX3] 259: ;Lightpenadresse lesen
[LX1] 260: mov dx,03b4h
[KD1] 261: mov al,16
[6H1] 262: out dx,al
[S11] 263: inc dx
[161] 264: in al,dx
[2F1] 265: mov ah,al
[L01] 266: dec dx
[MD1] 267: mov al,17
[6H1] 268: out dx,al
[S11] 269: inc dx
[161] 270: in al,dx
[BX4] 271: ;Mit Schwellwert vergleichen
[AT1] 272: cmp ax,2958
[KV0] 273: jb mda
[YU0] 274: herc:
[C77] 275: ;Grafik => Paker auf Bytezugriff umstel
len
[MD3] 276: mov byte ptr wort,0
[T82] 277: call vorspann
[B14] 278: mov si,offset hercules0
[ZC2] 279: call schreiben
[YT2] 280: ;BX := Seite * 32768
[PT2] 281: mov al,hercseite
[1A1] 282: mov ah,8
[321] 283: mul ah
[4F1] 284: mov bh,al
[8W1] 285: mov cx,32494
[9S1] 286: jmp packen
[000] 287:
[C84] 288: ;Grafikmodus einschalten
[ZV2] 289: hercules0 dw he-h0
[8G0] 290: h0:
[8S4] 291: ;Karte in Full-Modus schalten
[VZ1] 292: mov dx,03bfh
[8A1] 293: mov al,3
[6H1] 294: out dx,al
[151] 295: call h1
[G69] 296: ;Tabelle der Werte für den 6845 (Hi=Wer
t, Lo=RegisterNr)
[V05] 297: dw 3500h,2d01h,2e02h,0703h,5b04h,0205h
[315] 298: dw 5706h,5707h,0208h,0309h,000ah,000bh
[BG0] 299: h1:
[E31] 300: pop si
[5A3] 301: ;6845 initialisieren
[KE1] 302: mov cx,12
[LX1] 303: mov dx,03b4h
[EG0] 304: h2:
[PW0] 305: lodsw
[YH1] 306: out dx,ax
[H71] 307: loop h2
[8F5] 308: ;Controller auf Grafik schalten
[3Y1] 309: mov dx,03b8h
[6D1] 310: mov al,10
[6H1] 311: out dx,al
[8P0] 312: he = $
[000] 313:
[PF6] 314: ;Unterprogramm für EGA/VGA mit 16 Farbe
n
[XM0] 315: ega:
[T82] 316: call vorspann
[KX7] 317: ;Im monochromen Modus 0Fh gibt es keine
Palette

```



```

[363] 318: cmp neuermodus,0fh
[3A2] 319: je zweiteebene
[604] 320: mov si,offset egafarben
[2C2] 321: call schreiben
[552] 322: zweiteebene:
[322] 323: mov si,offset ega0
[2C2] 324: call schreiben
[4U5] 325: ;Indexregister auf Read-Map-Select
[U21] 326: mov dx,03ceh
[9A1] 327: mov al,4
[6H1] 328: out dx,al
[K62] 329: ;Lesen Seite 0
[S11] 330: inc dx
[5A1] 331: mov al,0
[6H1] 332: out dx,al
[TY1] 333: call packen
[849] 334: ;Der monochrome Modus verwendet nur die
Seiten 0 und 2
[363] 335: cmp neuermodus,0fh
[BP2] 336: je ueberspringel
[622] 337: mov si,offset egal
[2C2] 338: call schreiben
[N62] 339: ;Lesen Seite 1
[6A1] 340: mov al,1
[6H1] 341: out dx,al
[TY1] 342: call packen
[2F2] 343: ueberspringel:
[VX9] 344: ;Der EVA-Spezialmodus 800*600*4 verwend
et nur die Seiten 0 und 1
[B53] 345: cmp neuermodus,29h
[DP2] 346: je ueberspringe3
[922] 347: mov si,offset ega2
[2C2] 348: call schreiben
[S62] 349: ;Lesen Seite 2
[7A1] 350: mov al,2
[6H1] 351: out dx,al
[TY1] 352: call packen
[363] 353: cmp neuermodus,0fh
[DP2] 354: je ueberspringe3
[C22] 355: mov si,offset ega3
[2C2] 356: call schreiben
[V62] 357: ;Lesen Seite 3
[8A1] 358: mov al,3
[6H1] 359: out dx,al
[TY1] 360: call packen
[6F2] 361: ueberspringe3:
[DE1] 362: and cx,cx
[PY1] 363: jne egaende
[853] 364: cmp neuermodus,37h
[PY1] 365: jne egaende
[9VB] 366: ;Beim EVA-Spezialmodus 1024*768*16 ist
eine Seite 96 KByte groß und kann
[4B7] 367: ;nicht in einem Rutsch abgespeichert we
rden
[NC3] 368: mov videobasis,0b000h
[9W1] 369: mov cx,32768
[LT3] 370: mov si,offset evahires
[2C2] 371: call schreiben
[6U2] 372: jmp zweiteebene
[YA1] 373: egaende:
[3P0] 374: ret
[000] 375:
[DT2] 376: ;Palette setzen
[LM3] 377: egafarben dw ega0-egaf0
[RX0] 378: egaf0:
[GK1] 379: call egaf1
[CF3] 380: egapalette db 17 dup (0)
[TX0] 381: egaf1:
[X31] 382: pop dx
[EC1] 383: push es
[8C1] 384: push cs
[G31] 385: pop es
[AV1] 386: mov ax,1002h
[Y71] 387: int 10h
[G31] 388: pop es
[SC2] 389: ega0 dw egal-e0
[2G0] 390: e0:
[5V4] 391: ;Indexregister auf Map-Select
[PK1] 392: mov dx,03c4h
[7A1] 393: mov al,2
[6H1] 394: out dx,al
[DX2] 395: ;Schreiben Seite 0
[S11] 396: inc dx
[6A1] 397: mov al,1
[6H1] 398: out dx,al
[YC2] 399: egal dw ega2-e1
[5G0] 400: e1:
[GX2] 401: ;Schreiben Seite 1

[7A1] 402: mov al,2
[6H1] 403: out dx,al
[4D2] 404: ega2 dw ega3-e2
[8G0] 405: e2:
[LX2] 406: ;Schreiben Seite 2
[9A1] 407: mov al,4
[6H1] 408: out dx,al
[U63] 409: ega3 dw evahires-e3
[BG0] 410: e3:
[PK2] 411: ;Schreiben Seite 3
[DA1] 412: mov al,8
[6H1] 413: out dx,al
[0A3] 414: evahires dw egae-eh
[AK0] 415: eh:
[CR7] 416: ;Für EVA 1024*768*16 Bildschirmsegment
neu laden
[XZ1] 417: mov ax,0b000h
[3H1] 418: mov es,ax
[321] 419: egae = $
[000] 420:
[0D5] 421: ;Unterprogramm für VGA-Modus 13h
[2N0] 422: vga:
[T82] 423: call vorspann
[MR2] 424: call vgapalette
[9S1] 425: jmp packen
[000] 426:
[DP6] 427: ;VGA-Palette lesen und auf Disk schreib
en
[202] 428: vgapalette:
[713] 429: mov si,offset vga0
[2C2] 430: call schreiben
[TC1] 431: push bx
[WC1] 432: push cx
[LH1] 433: xor bx,bx
[412] 434: schreibedac:
[LZ6] 435: ;BIOS-Funktion: lese Palettenregister B
X
[TV1] 436: mov ax,1015h
[Y71] 437: int 10h
[U12] 438: ;Rot-Anteil
[9F1] 439: mov al,dh
[8N2] 440: call schreibeaal
[942] 441: ;Grün-Anteil
[8F1] 442: mov al,ch
[8N2] 443: call schreibeaal
[R02] 444: ;Blau-Anteil
[GF1] 445: mov al,cl
[8N2] 446: call schreibeaal
[G01] 447: inc bl
[WM2] 448: jne schreibedac
[V31] 449: pop cx
[T31] 450: pop bx
[A13] 451: mov si,offset vgal
[A82] 452: jmp schreiben
[000] 453:
[6F3] 454: ;VGA-Palette setzen
[FG2] 455: vga0 dw vgal-v0
[TD3] 456: v0 db 0e8h,0,3 ; call $+768
[8H2] 457: vgal dw vgae-v1
[7H0] 458: v1:
[X31] 459: pop dx
[EC1] 460: push es
[8C1] 461: push cs
[G31] 462: pop es
[DV1] 463: mov ax,1012h
[PKA] 464: ;BIOS-Funktion: Setze Block von Palette
nregistern mit [ES:DX]
[LH1] 465: xor bx,bx
[VL1] 466: mov cx,256
[Y71] 467: int 10h
[G31] 468: pop es
[531] 469: vgae = $
[000] 470:
[PD8] 471: ;Unterprogramm für die beiden EVA-Modi
mit 256 Farben
[AP0] 472: eva:
[T82] 473: call vorspann
[MR2] 474: call vgapalette
[K22] 475: mov si,offset eva0
[2C2] 476: call schreiben
[K62] 477: ;Lesen Seite 0
[PZ1] 478: mov dx,03cdh
[G7A] 479: ;Bit 0..2 Schreibseite, Bit 3..5 Lesese
ite, Bit 6 immer gesetzt
[1M1] 480: mov al,40h
[6H1] 481: out dx,al
[TY1] 482: call packen
[NZ2] 483: mov si,offset eval

```

Listing 1. Der Quellcode des Programms »snapshot.asm«


```

[ZC2] 484: call schreiben
[N62] 485: ;Lesen Seite 1
[HM1] 486: mov al,48h
[6H1] 487: out dx,al
[TY1] 488: call packen
[S22] 489: mov si,offset eva2
[ZC2] 490: call schreiben
[S62] 491: ;Lesen Seite 2
[2M1] 492: mov al,50h
[6H1] 493: out dx,al
[TY1] 494: call packen
[V22] 495: mov si,offset eva3
[ZC2] 496: call schreiben
[V62] 497: ;Lesen Seite 3
[KM1] 498: mov al,58h
[6H1] 499: out dx,al
[163] 500: cmp neuermodus,2dh
[TN1] 501: jne kein2d
[CW1] 502: mov cx,27392
[9S1] 503: jmp packen
[F71] 504: kein2d:
[TY1] 505: call packen
[Y22] 506: mov si,offset eva4
[ZC2] 507: call schreiben
[Y62] 508: ;Lesen Seite 4
[3M1] 509: mov al,60h
[6H1] 510: out dx,al
[363] 511: cmp neuermodus,2eh
[WN1] 512: jne kein2e
[NV1] 513: mov cx,45056
[9S1] 514: jmp packen
[K71] 515: kein2e:
[3P0] 516: ret
[000] 517:
[UN2] 518: eva0 dw eval-ev0
[6M0] 519: ev0:
[DX2] 520: ;Schreiben Seite 0
[P21] 521: mov dx,03cdh
[1M1] 522: mov al,40h
[6H1] 523: out dx,al
[XN2] 524: eval dw eva2-ev1
[AM0] 525: ev1:
[GX2] 526: ;Schreiben Seite 1
[3M1] 527: mov al,41h
[6H1] 528: out dx,al
[0P2] 529: eva2 dw eva3-ev2
[EM0] 530: ev2:
[LX2] 531: ;Schreiben Seite 2
[5M1] 532: mov al,42h
[6H1] 533: out dx,al
[3P2] 534: eva3 dw eva4-ev3
[KM0] 535: ev3:
[PX2] 536: ;Schreiben Seite 3
[7M1] 537: mov al,43h
[6H1] 538: out dx,al
[PP2] 539: eva4 dw evae-ev4
[PM0] 540: ev4:
[TX2] 541: ;Schreiben Seite 4
[9M1] 542: mov al,44h
[6H1] 543: out dx,al
[G31] 544: evae = $
[000] 545:
[6L5] 546: ;Daten für den Header der EXE-Datei
[PD3] 547: exeheader dw exee-exe
[PY4] 548: exe db 77,90,0,0,0,0,0,0,32,0,0,0,-1,-1,0
[5R3] 549: db 0,64,0,0,0,0,0,0,4,0,30,0,0,0,1
[641] 550: exee = $
[000] 551:
[UY7] 552: ;Die folgende Routine zeigt später das
Bild an
[MK6] 553: vorspanndat dw vorspannende-differenz
[5S1] 554: differenz:
[UL0] 555: cld
[BC1] 556: push ds
[8C1] 557: push cs
[E31] 558: pop ds
[683] 559: jmp short modusein
[VP7] 560: ;Auf Tastendruck warten, wenn Option "c
" fehlt
[321] 561: taste:
[E31] 562: pop ds
[R64] 563: cmp word ptr ds:[5dh]," C"
[1C2] 564: je.nichtwarten
[XK3] 565: ;Cursor ausschalten
[6V1] 566: mov cx,1000h
[LL1] 567: mov ah,01h
[Y71] 568: int 10h

```

```

[XU2] 569: ;Auf Taste warten
[T91] 570: mov ah,0
[G81] 571: int 16h
[355] 572: ;Videomodus wiederherstellen
[9B1] 573: mov ax,0
[7Y3] 574: altermodus = byte ptr $-2
[Y71] 575: int 10h
[XN2] 576: ;Bild verlassen
[R52] 577: nichtwarten:
[VM1] 578: mov ah,4ch
[381] 579: int 21h
[2F6] 580: ;Eine Seite (maximal 64 KByte) entpacke
n
[US1] 581: entpacken:
[EF1] 582: xor di,di
[X91] 583: mov ch,0
[PH1] 584: entpack0:
[NLC] 585: ;Quelldaten können unter widrigen Umstä
nden größer als 64 KByte sein,
;deshalb muß das Quellsegment bei Bedarf
f erhöht werden
[ET1] 587: cmp si,8000h
[3V1] 588: jb entpack1
[ET1] 589: sub si,8000h
[3H1] 590: mov ax,ds
[GR1] 591: add ax,0800h
[1H1] 592: mov ds,ax
[XK3] 593: ;Neue Anweisung lesen
[GH1] 594: entpack1:
[CV0] 595: lodsb
[KF1] 596: mov cl,al
[4H1] 597: cmp cl,128
[Y44] 598: ;Positiv: CX Werte übertragen
[7V1] 599: jb entpack2
[EW2] 600: ;128: Seite fertig
[MV1] 601: je entpack3
[S48] 602: ;Negativ: Folgender Wert wiederholt sic
h - CX mal
[6H0] 603: w0:
[CV0] 604: lodsb
[H01] 605: neg cl
[3PA] 606: ;Rep-Präfix wegen des variablen Byte-/W
ort-Zugriffs als Datum
[321] 607: db 0f3h
[9H0] 608: w1:
[NX0] 609: stosb
[P42] 610: jmp entpack0
[5P2] 611: ;Werte kopieren
[HH1] 612: entpack2:
[321] 613: db 0f3h
[CH0] 614: w2:
[PX0] 615: movsb
[P42] 616: jmp entpack0
[G52] 617: ;Seite fertig
[XH1] 618: entpack3:
[SN6] 619: ;DS für nächste Seite normalisieren
[RF1] 620: mov di,si
[DA1] 621: mov cl,4
[ME1] 622: shr di,cl
[3H1] 623: mov ax,ds
[EC1] 624: add ax,di
[1H1] 625: mov ds,ax
[9E7] 626: ;Sprung zur ersten Adresse hinter den D
aten
[PC1] 627: push ax
[8B1] 628: and si,15
[GC1] 629: push si
[PA1] 630: ;Ret-Far
[A31] 631: db 0cbh
[WA3] 632: ;Alten Modus retten
[7K1] 633: modusein:
[XM1] 634: mov ah,0fh
[Y71] 635: int 10h
[FF5] 636: ;Variablenzugriff zur Laufzeit!
[SN5] 637: mov ds:[altermodus-differenz],al
[Z34] 638: ;Neuen Modus einschalten
[9B1] 639: mov ax,0
[GY3] 640: neuermodus = byte ptr $-2
[Y71] 641: int 10h
[AH3] 642: ;Zielsegment setzen
[9B1] 643: mov ax,0
[LU3] 644: videobasis = word ptr $-2
[3H1] 645: mov es,ax
[8C1] 646: push cs
[1M2] 647: vorspannende = $
[000] 648:
[4C8] 649: ;Programmblock zwischen den einzelnen B
ildseiten

```



```

[K83] 650: neueebene dw nee-ne0
[5L0] 651: ne0:
[P94] 652: ;SI auf Datenblock setzen
[M91] 653: call nel
[9L0] 654: nel:
[E31] 655: pop si
[N32] 656: add si,nee-nel
[VM5] 657: ;Adresse des Entpackers auf Stack
[R31] 658: pop ax
[PC1] 659: push ax
[PC1] 660: push ax
[ZM4] 661: mov ax,entpacken-differenz
[PC1] 662: push ax
[W91] 663: ;Ret Far
[A31] 664: db 0cbh
[901] 665: nee = $
[000] 666:
[2D4] 667: ;Sprung zum Programmende
[9A3] 668: nachspann dw nse-ns
[XL0] 669: ns:
[ZX3] 670: mov ax,taste-differenz
[PC1] 671: push ax
[A31] 672: db 0cbh
[L11] 673: nse = $
[000] 674:
[ZP3] 675: ;Flag für Wortzugriff
[BE1] 676: word db 0
[7A5] 677: ;Adressen der Byte-/Wort-Befehle
[A26] 678: worte dw w0,w1,w2,w3,w4,w5,w6,w7,w8,w9,0
[4S1] 679: handle dw ?
[000] 680:
[KS4] 681: ;Entpacker auf Disk schreiben
[HM1] 682: vorspann:
[XL6] 683: ;Befehle an Byte-/Wortzugriff anpassen
[NE3] 684: mov si,offset worte
[3B1] 685: push di
[M42] 686: neuerbefehl:
[PW0] 687: lodsw
[7E1] 688: and ax,ax
[BS2] 689: ;Adresse=0 => Ende
[A22] 690: je befehlende
[3G1] 691: mov di,ax
[6G2] 692: mov al,ds:[di]
[2S8] 693: ;Bit 0 des Befehls ist für Wortzugriff
verantwortlich
[XU1] 694: and al,0feh
[WN1] 695: or al,wort
[CH2] 696: mov ds:[di],al
[NT2] 697: jmp neuerbefehl
[5Y1] 698: befehlende:
[G21] 699: pop di
[UD4] 700: mov si,offset vorspanndat
[ZN6] 701: ;Datenblock [DS:SI] auf Disk schreiben
[XP1] 702: schreiben:
[WC1] 703: push cx
[PW0] 704: lodsw
[EH1] 705: mov cx,ax
[8D2] 706: jcxz datenende
[722] 707: neuesdatum:
[CV0] 708: lodsb
[8N2] 709: call schreibbeal
[LU2] 710: loop neuesdatum
[ZR1] 711: datenende:
[V31] 712: pop cx
[3P0] 713: ret
[000] 714:
[L9A] 715: ;CX Bytes ab [VideoBasis:BX] gepackt au
f Diskette schreiben
[P81] 716: packen:
[PK5] 717: ;Aufruf des Entpackers schreiben
[A24] 718: mov si,offset neueebene
[ZC2] 719: call schreiben
[TC1] 720: push bx
[WC1] 721: push cx
[ZC1] 722: push dx
[BC1] 723: push ds
[9Y2] 724: mov ds,videobasis
[2H1] 725: mov si,bx
[ED1] 726: add cx,bx
[49A] 727: ;Endevergleich beim 1. Mal überspringen
(für 64 KByte-Blöcke)
[UP3] 728: jmp short ersterblock
[W12] 729: neuerblock:
[LL2] 730: ;Ende erreicht?
[HF1] 731: cmp si,cx
[BV1] 732: je packende
[162] 733: ersterblock:
[GC1] 734: push si

[SZ4] 735: ;Bytezähler initialisieren
[BA1] 736: mov dl,0
[FH0] 737: w3:
[W63] 738: ;Ersten Wert holen
[CV0] 739: lodsb
[CH1] 740: mov bx,ax
[0H3] 741: jmp short pruefeende
[KH0] 742: w4:
[2R3] 743: ;Nächsten Wert holen
[CV0] 744: lodsb
[VR1] 745: xchg ax,bx
[NH0] 746: w5:
[705] 747: ;Mit vorigem Wert vergleichen
[YD1] 748: cmp bl,al
[8C1] 749: je gleich
[M01] 750: inc dl
[5LC] 751: ;Maximal 127 verschiedene Werte können
an einem Stück verarbeitet werden
[3H1] 752: cmp dl,127
[932] 753: je schreibedl
[B12] 754: pruefeende:
[HF1] 755: cmp si,cx
[WZ0] 756: jne w4
[M01] 757: inc dl
[R96] 758: ;Zähler und Werte auf Disk schreiben
[7Z1] 759: schreibedl:
[E31] 760: pop si
[HF1] 761: mov al,dl
[8N2] 762: call schreibbeal
[SH0] 763: w6:
[CV0] 764: lodsb
[SP2] 765: call schreibeax
[FZ0] 766: dec dl
[201] 767: jne w6
[FH2] 768: jmp neuerblock
[N81] 769: gleich:
[NL7] 770: ;Falls Zähler > 0, dann zuerst Werte sc
hreiben
[T81] 771: cmp dl,0
[HD2] 772: jne schreibedl
[2G5] 773: ;Zwei Werte sind bereits gleich
[CD1] 774: mov dl,-2
[6B2] 775: pruefefertig:
[HF1] 776: cmp si,cx
[TW1] 777: je ungleich
[VH0] 778: w7:
[2R3] 779: ;Nächsten Wert holen
[CV0] 780: lodsb
[YH0] 781: w8:
[D54] 782: ;Auf Gleichheit prüfen
[XD1] 783: cmp al,bl
[732] 784: jne ungleich
[FZ0] 785: dec dl
[1DB] 786: ;Maximal 127 gleiche Werte können an ei
nem Stück verarbeitet werden
[AR1] 787: cmp dl,-127
[4X2] 788: jne pruefefertig
[9H1] 789: ungleich:
[GY9] 790: ;Zähler und sich wiederholender Wert au
f Disk schreiben
[HF1] 791: mov al,dl
[8N2] 792: call schreibbeal
[E31] 793: pop si
[1K0] 794: w9:
[CV0] 795: lodsb
[M01] 796: inc dl
[B01] 797: jne w9
[SP2] 798: call schreibeax
[FH2] 799: jmp neuerblock
[RF1] 800: packende:
[FH4] 801: ;Endekennzeichen schreiben
[KK1] 802: mov al,128
[8N2] 803: call schreibbeal
[E31] 804: pop ds
[X31] 805: pop dx
[V31] 806: pop cx
[T31] 807: pop bx
[3P0] 808: ret
[000] 809:
[KY8] 810: ;Byte oder Wort in AX schreiben, je nac
h Wert in "Wort"
[502] 811: schreibeax:
[4P2] 812: cmp cs:[wort],0
[632] 813: je schreibbeal
[PC1] 814: push ax
[8N2] 815: call schreibbeal
[R31] 816: pop ax
[6F1] 817: mov al,ah

```

Listing 1. Der Quellcode des Programms »snapshot.asm«


```
[FX3] 818: ;AL in Puffer schreiben
[1Z1] 819: schreibbeal:
[NX0] 820: stosb
[PA2] 821: ;Puffer voll?
[YG1] 822: cmp di,256
[5K2] 823: jb schreibeende
[6K4] 824: ;Puffer auf Disk schreiben
[YD2] 825: schreibeblock:
[5C1] 826: and di,di
[6K2] 827: je schreibeende
[TC1] 828: push bx
[WC1] 829: push cx
[ZC1] 830: push dx
[BC1] 831: push ds
[GS2] 832: mov bx,cs:handle
[LG1] 833: mov cx,di
[6C1] 834: push cs
[E31] 835: pop ds
[TH1] 836: xor dx,dx
[NL1] 837: mov ah,40h
[381] 838: int 21h
[E31] 839: pop ds
[X31] 840: pop dx
[V31] 841: pop cx
[T31] 842: pop bx
[DA6] 843: ;Bei Fehlern ist das Carryflag gesetzt
[R52] 844: jc schliessen
[C84] 845: ;Prüfen ob Diskette voll
[XE1] 846: cmp ax,di
[1G2] 847: jne schliessen
[TT4] 848: ;Pufferzeiger zurücksetzen
[EF1] 849: xor di,di
[G62] 850: schreibeende:
[3P0] 851: ret
[R12] 852: schliessen:
[N62] 853: mov bx,handle
[YM1] 854: mov ah,3eh
[381] 855: int 21h
[F91] 856: fehler:
[SA1] 857: ;Fiepen
[7X1] 858: mov ax,0e07h
[Y71] 859: int 10h
[W65] 860: ;Rückkehr zum Hauptprogramm
[7H1] 861: mov sp,bp
[3P0] 862: ret
[000] 863:
[3K2] 864: hauptprogramm:
[D97] 865: ;Stackpointer für Fehlerrückkehr sicher
n
[PG1] 866: mov bp,sp
[5W2] 867: mov al,usermodus
[AV2] 868: mov bh,userseite
[3E5] 869: ;Moduserkennung auf Automatik?
[P74] 870: cmp byte ptr automodus,0
[HN1] 871: je manuell
[P86] 872: ;Aktuellen Videomodus und Seite holen
[XM1] 873: mov ah,0fh
[Y71] 874: int 10h
[ZE1] 875: manuell:
[LV4] 876: ;Videomodus in Tabelle suchen
[544] 877: mov si,offset videomodi
[952] 878: vergleichen:
[3F2] 879: cmp al,ds:[si]
[RX1] 880: je gefunden
[G23] 881: add si,lengthvideo
[T14] 882: cmp si,offset videoende
[5A2] 883: jb vergleichen
[2T1] 884: jmp fehler
[8G1] 885: gefunden:
[U13] 886: mov neuermodus,al
[CD5] 887: ;Datei öffnen, Attribut: Archive
[3M1] 888: mov cl,20h
[T23] 889: mov dx,offset pfad
[UM1] 890: mov ah,3ch
[381] 891: int 21h
[P62] 892: mov handle,ax
[X72] 893: jnc oeffnenok
[1A1] 894: fehler:
[2T1] 895: jmp fehler
[1S1] 896: oeffnenok:
[8T8] 897: ;Pufferzeiger auf 0 setzen, Puffer ist
nutzloser PSP
[EF1] 898: xor di,di
[GC1] 899: push si
[K14] 900: ;EXE-Dateikopf schreiben
[S14] 901: mov si,offset exeheader
[ZC2] 902: call schreiben
[E31] 903: pop si
```

```
[RC7] 904: ;Unrelevante Werte im EXE-Kopf mit 0 fü
llen
[ZL1] 905: mov cx,547
[Y82] 906: schreibnull:
[5A1] 907: mov al,0
[8N2] 908: call schreibbeal
[313] 909: loop schreibnull
[1Y3] 910: ;Videomodus überlesen
[CV0] 911: lodsb
[H54] 912: ;Bildschirmsegment lesen
[PW0] 913: lodsw
[2Y2] 914: mov videobasis,ax
[ZE8] 915: ;Seitenfaktor mit aktueller Seite multi
plizieren
[CV0] 916: lodsb
[521] 917: mul bh
[7A1] 918: mov bl,0
[LH3] 919: ;und mit 256 malnehmen
[4F1] 920: mov bh,al
[TN3] 921: ;Bytes pro Seite holen
[PW0] 922: lodsw
[EH1] 923: mov cx,ax
[DU4] 924: ;Flag für Wortzugriff holen
[CV0] 925: lodsb
[142] 926: mov wort,al
[X74] 927: ;Unterprogramm aufrufen
[PW0] 928: lodsw
[891] 929: call ax
[D44] 930: mov si,offset nachspann
[ZC2] 931: call schreiben
[H28] 932: ;Eventuelle Werte im Puffer auf Disk sc
hreiben
[133] 933: call schreibeblock
[9U4] 934: ;Dateilänge nach DX:AX holen,
[HV1] 935: mov ax,4202h
[N62] 936: mov bx,handle
[PH1] 937: xor cx,cx
[TH1] 938: xor dx,dx
[381] 939: int 21h
[EH1] 940: jc fehler0
[P64] 941: ;ins EXE-Format wandeln,
[2F1] 942: add ax,511
[A71] 943: adc dx,0
[AL1] 944: mov cx,512
[931] 945: div cx
[ZY0] 946: stosw
[LV6] 947: ;und auf Position 4 in der Datei schrei
ben
[9V1] 948: mov ax,4200h
[PH1] 949: xor cx,cx
[LB1] 950: mov dx,4
[381] 951: int 21h
[EH1] 952: jc fehler0
[133] 953: call schreibeblock
[S13] 954: ;Datei schließen
[N62] 955: mov bx,handle
[YM1] 956: mov ah,3eh
[381] 957: int 21h
[0W2] 958: jnc schliessenok
[2T1] 959: jmp fehler
[AA2] 960: schliessenok:
[KL4] 961: ;Ende des Dateinamens suchen
[A23] 962: mov si,offset pfad
[LU1] 963: suchenull:
[CV0] 964: lodsb
[L81] 965: cmp al,0
[RA2] 966: jne suchenull
[KN0] 967: std
[CV0] 968: lodsb
[8V5] 969: ;Rückwärts nach Ziffern suchen
[LB2] 970: pruefeziffer:
[S03] 971: cmp si,offset pfad
[V41] 972: jb ende
[CV0] 973: lodsb
[P95] 974: ;Abbruch bei einem Subdirectory
[3H1] 975: cmp al,"\"
[451] 976: je ende
[BE1] 977: cmp al,"0"
[CP2] 978: jb pruefeziffer
[XE1] 979: cmp al,"9"
[9P2] 980: ja pruefeziffer
[BB5] 981: ;Gefundene Ziffer weiterzählen
[E01] 982: inc al
[FN2] 983: mov ds:[si+1],al
[XE1] 984: cmp al,"9"
[RA1] 985: jbe ende
[M78] 986: ;Bei Überlauf von "9" auf "0" nächste Z
iffer suchen
```



```

[PC4] 987: mov byte ptr ds:[si+1], "0"
[UY2] 988: jmp pruefezeiffer
[XT0] 989: ende:
[3P0] 990: ret
[000] 991:
[TM6] 992: ;Neue bzw. gerettete Interruptvektoren
[084] 993: alterint05 dw neuerint05,?
[K74] 994: alterint10 dw neuerint10,?
[P84] 995: alterint19 dw neuerint19,?
[G84] 996: alterint09 dw neuerint09,?
[C84] 997: alterint08 dw neuerint08,?
[S84] 998: alterint28 dw neuerint28,?
[884] 999: alterint24 dw neuerint24,?
[X4A] 1000: ;Zeiger auf Flag, das den Aufruf von DO
;S-Funktionen erlaubt
[5G2] 1001: indosflag dw ?,?
[UL5] 1002: ;Verschiedene Schalterzustände
[X82] 1003: usermodus db 0
[282] 1004: userseite db 0
[FF2] 1005: automodus db -1
[Y72] 1006: hercmodus db 0
[372] 1007: hercseite db 0
[556] 1008: ;Flag für ALT gedrückt (nur beim PC)
[HW1] 1009: altflag db 0
[X54] 1010: ;Flag für PrtSc verboten
[AH1] 1011: aktiv db 0
[000] 1012:
[0X2] 1013: ;Hotkey erkennen
[P21] 1014: neuerint05:
[4X2] 1015: assume ds:nothing
[M12] 1016: if intr eq 9
[S24] 1017: ;Routine im Ticker-Interrupt
[PC1] 1018: push ax
[BC1] 1019: push ds
[GH1] 1020: xor ax,ax
[1H1] 1021: mov ds,ax
[D0B] 1022: ;Prüfen, ob der Tastaturinterrupt 09h n
;och auf das Programm zeigt
[2H1] 1023: mov ax,cs
[F33] 1024: cmp ds:[09h*4+2],ax
[1P2] 1025: je ueberwachraus
[HM9] 1026: ;Falls nicht, dann sich wieder an die e
;rste Stelle vormogeln
[P43] 1027: xchg ax,ds:[09h*4+2]
[9B3] 1028: mov alterint09+2,ax
[R44] 1029: mov ax,offset neuerint09
[703] 1030: xchg ax,ds:[09h*4]
[HX2] 1031: mov alterint09,ax
[9F2] 1032: ueberwachraus:
[E31] 1033: pop ds
[R31] 1034: pop ax
[V24] 1035: jmp dword ptr alterint05
[TU0] 1036: endif
[000] 1037:
[202] 1038: neuerint09:
[PC1] 1039: push ax
[M12] 1040: if intr eq 9
[B56] 1041: ;Scan-Cde der gedrückten Taste lesen
[H71] 1042: in al,60h
[2E3] 1043: ;Wurde Alt gedrückt?
[YK1] 1044: cmp al,38h
[0G2] 1045: jne keinaltein
[BC2] 1046: mov altflag,-1
[K23] 1047: jmp short int09alt
[P12] 1048: keinaltein:
[V23] 1049: ;Wurde Alt losgelassen?
[ZU1] 1050: cmp al,0B8h
[3H2] 1051: jne keinaltaus
[Y72] 1052: mov altflag,0
[K23] 1053: jmp short int09alt
[T22] 1054: keinaltaus:
[SK1] 1055: cmp al,54h
[Y82] 1056: je sysrequest
[DF3] 1057: ;Ist die Taste PrtSc?
[WK1] 1058: cmp al,37h
[732] 1059: jne int09alt
[886] 1060: ;Ja: Ist gleichzeitig Alt gedrückt?
[C62] 1061: cmp altflag,0
[VE6] 1062: ;Nein, dann zur alten Tastaturroutine
[LV1] 1063: je int09alt
[F72] 1064: sysrequest:
[TU0] 1065: endif
[7B6] 1066: ;Abbruch, wenn Programm bereits aktiv
[TX1] 1067: cmp aktiv,0
[TG1] 1068: jne adios
[G02] 1069: mov aktiv,-1
[GCI] 1070: push si
[3B1] 1071: push di

[0G6] 1072: ;Ticker- und Dosfree-Interrupt setzen
[MG2] 1073: call int08und28
[G21] 1074: pop di
[E31] 1075: pop si
[N01] 1076: adios:
[C12] 1077: if intr eq 5
[R31] 1078: pop ax
[GR0] 1079: iret
[ZR0] 1080: else
[S35] 1081: ;Empfangene Taste bestätigen
[K71] 1082: in al,61h
[R61] 1083: or al,128
[YN1] 1084: out 61h,al
[CF1] 1085: and al,127
[YN1] 1086: out 61h,al
[696] 1087: ;Interruptausführung bestätigen
[5M1] 1088: mov al,61h
[LN1] 1089: out 20h,al
[R31] 1090: pop ax
[GR0] 1091: iret
[9H1] 1092: int09alt:
[R31] 1093: pop ax
[Z24] 1094: jmp dword ptr alterint09
[TU0] 1095: endif
[000] 1096:
[LT5] 1097: ;Standardpalette für EGA und VGA
[4E7] 1098: egastandard db 0,1,2,3,4,5,20,7,56,57,58,59
;60,61,62,63,0
[000] 1099:
[RP3] 1100: ;Neuer Videointerrupt
[9Z1] 1101: neuerint10:
[RB1] 1102: and ah,ah
[ME2] 1103: je modussetzen
[3L1] 1104: cmp ah,0bh
[WW1] 1105: je cgafarben
[0K1] 1106: cmp ah,10h
[3C2] 1107: jae egadervga
[LK1] 1108: raushier:
[U24] 1109: jmp dword ptr alterint10
[3X4] 1110: ;Neuen Videomodus einschalten
[872] 1111: modussetzen:
[UD7] 1112: ;CGA- und EGA-Farben auf Standartwerte
;setzen
[482] 1113: mov blinken,1
[WA2] 1114: mov cgarand,16
[LP2] 1115: mov cgapalette,0
[PC1] 1116: push ax
[TC1] 1117: push bx
[SE1] 1118: mov bx,16
[5F2] 1119: naechstefarbe:
[064] 1120: mov al,[egastandard+bx]
[E14] 1121: mov [egapalette+bx],al
[BZ0] 1122: dec bl
[113] 1123: jns naechstefarbe
[T31] 1124: pop bx
[R31] 1125: pop ax
[U52] 1126: jmp raushier
[RP5] 1127: ;CGA-Randfarbe bzw. Palette setzen
[MN1] 1128: cgafarben:
[981] 1129: cmp bh,0
[C32] 1130: jne keinrand
[0E2] 1131: mov cgarand,b1
[U52] 1132: jmp raushier
[Eh1] 1133: keinrand:
[XV2] 1134: mov cgapalette,b1
[U52] 1135: jmp raushier
[0Z1] 1136: egadervga:
[GT1] 1137: cmp ax,1000h
[T32] 1138: jne overscan
[216] 1139: ;Einzelnes Palettenregister setzen
[4C1] 1140: cmp cl,16
[A32] 1141: jae raushier
[3B1] 1142: push di
[5G1] 1143: mov di,cx
[AF1] 1144: and di,255
[VZ3] 1145: mov [egapalette+di],bh
[G21] 1146: pop di
[U52] 1147: jmp raushier
[VH1] 1148: overscan:
[MT1] 1149: cmp ax,1001h
[882] 1150: jne farblock
[3V2] 1151: ;Randfarbe setzen
[M83] 1152: mov egapalette+16,bh
[U52] 1153: jmp raushier
[4S1] 1154: farblock:
[ST1] 1155: cmp ax,1002h
[D92] 1156: jne blinktest
[RG5] 1157: ;Komplette Farbpalette neu setzen

```

Listing 1. Der Quellcode des Programms »snapshot.asm«


```

[PC1] 1158: push ax
[TC1] 1159: push bx
[3B1] 1160: push di
[6G1] 1161: mov di,dx
[SE1] 1162: mov bx,16
[NS1] 1163: neuefarbe:
[BW2] 1164: mov al,es:[di+bx]
[E14] 1165: mov [egapalette+bx],al
[BZ0] 1166: dec bl
[HA2] 1167: jns neuefarbe
[G21] 1168: pop di
[T31] 1169: pop bx
[R31] 1170: pop ax
[U52] 1171: jmp raushier
[9T1] 1172: blinktest:
[WT1] 1173: cmp ax,1003h
[0E1] 1174: jne.raus
[AG2] 1175: mov blinken,bl
[2W0] 1176: raus:
[U52] 1177: jmp raushier
[000] 1178:
[HU6] 1179: ;Interrupts bei Warmstart deaktivieren
[402] 1180: neuerint19:
[CG2] 1181: call int05und19
[T81] 1182: int 19h
[000] 1183:
[DF8] 1184: ;Neuer Ticker-Interrupt bei aktiviertem
        Programm
[ZZ1] 1185: neuerint08:
[6P3] 1186: ;Alte Routine bedienen
[ZX0] 1187: pushf
[144] 1188: call dword ptr alterint08
[BC1] 1189: push ds
[GC1] 1190: push si
[GB4] 1191: ;Sind DOS-Aufrufe erlaubt?
[7H4] 1192: lds si,dword ptr indosflag
[DV3] 1193: cmp byte ptr ds:[si],0
[E31] 1194: pop si
[E31] 1195: pop ds
[SP1] 1196: jne geduld
[LD8] 1197: ;Neuer Dosfree-Interrupt bei aktivierte
        m Programm
[302] 1198: neuerint28:
[PC1] 1199: push ax
[TC1] 1200: push bx
[WC1] 1201: push cx
[ZC1] 1202: push dx
[GC1] 1203: push si
[3B1] 1204: push di
[TB1] 1205: push bp
[BC1] 1206: push ds
[EC1] 1207: push es
[SR7] 1208: ;Ticker- und Dosfree-Interrupt restauri
        eren
[MG2] 1209: call int08und28
[7T3] 1210: ;Interrupts zulassen
[6P0] 1211: sti
[0LB] 1212: ;Fehlerbehandlungsroutine für Schreibfe
        hler setzen (erst ab DOS 3.0)
[CL1] 1213: call int24
[2H1] 1214: mov ax,cs
[3H1] 1215: mov es,ax
[173] 1216: call hauptprogramm
[695] 1217: ;Fehlerbehandlung restaurieren
[CL1] 1218: call int24
[G31] 1219: pop es
[E31] 1220: pop ds
[131] 1221: pop bp
[G21] 1222: pop di
[E31] 1223: pop si
[X31] 1224: pop dx
[V31] 1225: pop cx
[T31] 1226: pop bx
[R31] 1227: pop ax
[KC4] 1228: ;Neuen Aufruf ermöglichen
[B21] 1229: mov aktiv,0
[E81] 1230: geduld:
[GR0] 1231: iret
[000] 1232:
[N84] 1233: ;Schreibfehlerbehandlung
[RZ1] 1234: neuerint24:
[W75] 1235: ;Code 3 entspricht (I)gnorieren
[8A1] 1236: mov al,3
[GR0] 1237: iret
[000] 1238:
[NY5] 1239: ;Hotkey-Erkennung ein-/ausschalten
[PY1] 1240: int05und19:
[C12] 1241: if intr eq 5

```

```

[0V1] 1242: mov di,05h*4
[ZR0] 1243: else
[6V1] 1244: mov di,08h*4
[TU0] 1245: endif
[V24] 1246: mov si,offset alterint05
[5H2] 1247: call interrupt
[RU1] 1248: mov di,10h*4
[U24] 1249: mov si,offset alterint10
[5H2] 1250: call interrupt
[9V1] 1251: mov di,19h*4
[334] 1252: mov si,offset alterint19
[WK3] 1253: jmp short interrupt
[RN6] 1254: ;Warten auf freies DOS ein-/ausschalten
[VY1] 1255: int08und28:
[BC1] 1256: push ds
[EC1] 1257: push es
[6V1] 1258: mov di,08h*4
[Y24] 1259: mov si,offset alterint08
[5H2] 1260: call interrupt
[8V1] 1261: mov di,28h*4
[634] 1262: mov si,offset alterint28
[5H2] 1263: call interrupt
[G31] 1264: pop es
[E31] 1265: pop ds
[3P0] 1266: ret
[ZT5] 1267: ;Fehlerbehandlung ein-/ausschalten
[M01] 1268: int24:
[0V1] 1269: mov di,24h*4
[Z34] 1270: mov si,offset alterint24
[BC9] 1271: ;Interruptvektor mit gesichertem Vektor
        vertauschen
[5X1] 1272: interrupt:
[UL0] 1273: cld
[2H1] 1274: mov ax,cs
[1H1] 1275: mov ds,ax
[GH1] 1276: xor ax,ax
[3H1] 1277: mov es,ax
[ZT2] 1278: call vertauschen
[S72] 1279: vertauschen:
[BH2] 1280: mov ax,es:[di]
[0Z0] 1281: movsw
[4P2] 1282: mov ds:[si-2],ax
[3P0] 1283: ret
[000] 1284:
[2W7] 1285: ;Dateiname beim ersten Aufruf ohne Name
        nsangabe
[G03] 1286: pfad db "BILD0.EXE",0
[000] 1287:
[HA2] 1288: installieren:
[FA2] 1289: assume ds:code
[7L4] 1290: mov si,offset programmname
[K51] 1291: call write
[0P9] 1292: ;Segmentadresse der neuen Interruptproz
        eduren setzen
[994] 1293: mov di,offset alterint05+2
[2H1] 1294: mov ax,cs
[MB1] 1295: mov cx,7
[6B2] 1296: setzesegment:
[ZY0] 1297: stosw
[361] 1298: add di,2
[523] 1299: loop setzesegment
[HA7] 1300: ;Prüfen, ob Programm schon installiert
        ist
[XX2] 1301: mov ax,3500h+intrn
[381] 1302: int 21h
[C12] 1303: if intrn eq 5
[A44] 1304: mov si,offset neuerint05
[ZR0] 1305: else
[E44] 1306: mov si,offset neuerint09
[TU0] 1307: endif
[4G1] 1308: mov di,bx
[ME1] 1309: mov cx,32
[GT1] 1310: repe cmpsw
[9L1] 1311: je schonda
[ZX4] 1312: mov si,offset installmeldung
[KS1] 1313: call write
[S55] 1314: ;Hotkey-Erkennung einschalten
[EM0] 1315: cli
[CG2] 1316: call int05und19
[505] 1317: ;Adresse des InDosFlags holen
[WL1] 1318: mov ah,34h
[381] 1319: int 21h
[NT2] 1320: mov indosflag,bx
[923] 1321: mov indosflag+2,es
[EP8] 1322: ;Schaltervariablen liegen im aktuellen
        Code-Segment
[8C1] 1323: push cs
[G31] 1324: pop es

```



```

[665] 1325:      ;Dateinamen und Schalter lesen
[4C1] 1326: schonda:
[DL1] 1327:      mov si,128
[CV0] 1328:      lodsb
[8W5] 1329:      ;Parameterstring mit 0 abschließen
[T91] 1330:      mov ah,0
[0D1] 1331:      add si,ax
[ZW3] 1332:      mov byte ptr ds:[si],0
[GL1] 1333:      mov si,129
[024] 1334:      ;Leerzeichen überlesen
[842] 1335: leerzeichen:
[CV0] 1336:      lodsb
[L81] 1337:      cmp al,0
[422] 1338:      je keinparam0
[BD1] 1339:      cmp al," "
[RA2] 1340:      je leerzeichen
[9E1] 1341:      cmp al,"/"
[3X1] 1342:      je schalter
[PZ3] 1343:      ;Dateinamen übertragen
[CL3] 1344:      mov di,offset pfad
[AG6] 1345:      ;Maximal 63 Zeichen mit Null-Abschluß
[7D1] 1346:      mov ah,64
[Y52] 1347: uebertragen:
[XY0] 1348:      dec ah
[BR1] 1349:      jne pfadok
[3G4] 1350:      mov si,offset laengefehler
[KS1] 1351:      call write
[NY1] 1352:      keinparam0:
[023] 1353:      jmp keinparameter
[191] 1354:      pfadok:
[6P7] 1355:      ;Der Name wird mit 0, Leerzeichen oder
      "/" beendet
[L81] 1356:      cmp al,0
[422] 1357:      je keinparam0
[9E1] 1358:      cmp al,"/"
[3X1] 1359:      je schalter
[BD1] 1360:      cmp al," "
[AU2] 1361:      je neuerschalter0
[ED7] 1362:      ;Kleinbuchstaben in Großbuchstaben wand
      eln
[DF1] 1363:      cmp al,"a"
[AM2] 1364:      jb nichtwandeln
[ZG1] 1365:      cmp al,"z"
[7M2] 1366:      ja nichtwandeln
[ST1] 1367:      and al,255-32
[L82] 1368:      nichtwandein:
[NX0] 1369:      stosb
[M74] 1370:      ;Namen mit 0 abschließen
[NV3] 1371:      mov byte ptr es:[di],0
[2L4] 1372:      ;Nächstes Zeichen prüfen
[CV0] 1373:      lodsb
[ZU2] 1374:      jmp uebertragen
[YH1] 1375:      schalter:
[CV0] 1376:      lodsb
[R56] 1377:      ;Schalter in Kleinbuchstaben wandeln
[F31] 1378:      or al,32
[DF1] 1379:      cmp al,"a"
[N13] 1380:      jne keinautomatik
[DP3] 1381:      mov es:[automodus],-1
[CS2] 1382:      neuerschalter0:
[F33] 1383:      jmp neuerschalter
[HH2] 1384:      keinautomatik:
[VF1] 1385:      cmp al,"h"
[503] 1386:      jne keinhercmodus
[RM3] 1387:      mov es:[hercmodus],-1
[0S5] 1388:      ;Ist eine Seite bei "/" angegeben?
[CV0] 1389:      lodsb
[3E1] 1390:      cmp al,"/"
[113] 1391:      jne sucheschalter
[CV0] 1392:      lodsb
[LY4] 1393:      ;Mögliche Seiten sind 0 und 1
[BE1] 1394:      cmp al,"0"
[3C2] 1395:      je hercseiteok
[DE1] 1396:      cmp al,"1"
[3C2] 1397:      je hercseiteok
[T03] 1398:      jmp falschoption
[M52] 1399:      hercseiteok:
[BE1] 1400:      sub al,"0"
[6P3] 1401:      mov es:[hercseite],al
[D74] 1402:      jmp short neuerschalter
[SP1] 1403:      schalter0:
[552] 1404:      jmp schalter
[1G2] 1405:      keinhercmodus:
[5G1] 1406:      cmp al,"m"
[HU2] 1407:      jne keinmdamodus
[SH3] 1408:      mov es:[hercmodus],0
[D74] 1409:      jmp short neuerschalter
[L82] 1410:      keinmdamodus:

```

```

[GC8] 1411:      ;Ist der Schalter eine gültige Hexadezi
      malziffer?
[PD2] 1412:      call hexziffer
[GV2] 1413:      jc falschoption0
[DA1] 1414:      mov cl,4
[ZD1] 1415:      shl al,cl
[PC1] 1416:      push ax
[L64] 1417:      ;Zweite Hexziffer holen
[CV0] 1418:      lodsb
[F31] 1419:      or al,32
[PD2] 1420:      call hexziffer
[V31] 1421:      pop cx
[GV2] 1422:      jc falschoption0
[961] 1423:      or al,cl
[5P5] 1424:      ;Ist der angegebene Modus bekannt?
[444] 1425:      mov bx,offset videomodi
[H11] 1426:      suche:
[HF2] 1427:      cmp al,es:[bx]
[RP2] 1428:      je modusbekannt
[V23] 1429:      add bx,lengthvideo
[424] 1430:      cmp bx,offset videoende
[581] 1431:      jb suche
[DC4] 1432:      mov si,offset modusfehler
[P44] 1433:      jmp short fehlerausgabe
[392] 1434:      modusbekannt:
[0S3] 1435:      mov es:[usermodus],al
[EL3] 1436:      mov es:[automodus],0
[3S6] 1437:      ;Ist außerdem noch eine Seite angegeben
      ?
[CV0] 1438:      lodsb
[3E1] 1439:      cmp al,"/"
[113] 1440:      jne sucheschalter
[CV0] 1441:      lodsb
[PD2] 1442:      call hexziffer
[PN2] 1443:      falschoption0:
[GS2] 1444:      jc falschoption
[5R3] 1445:      mov es:[userseite],al
[U26] 1446:      ;Folgt eine weitere Schalterangabe?
[2K2] 1447:      neuerschalter:
[CV0] 1448:      lodsb
[XG2] 1449:      sucheschalter:
[L81] 1450:      cmp al,0
[GP2] 1451:      je keinparameter
[BD1] 1452:      cmp al," "
[AR2] 1453:      je neuerschalter
[9E1] 1454:      cmp al,"/"
[LY1] 1455:      je schalter0
[PK4] 1456:      mov si,offset syntaxfehler
[TC2] 1457:      fehlerausgabe:
[KS1] 1458:      call write
[LG2] 1459:      keinparameter:
[N05] 1460:      ;Einschaltmeldungen ausgeben
[6P0] 1461:      sti
[RA4] 1462:      mov si,offset pfadmeldung
[KS1] 1463:      call write
[BC1] 1464:      push ds
[EC1] 1465:      push es
[E31] 1466:      pop ds
[A23] 1467:      mov si,offset pfad
[KS1] 1468:      call write
[E31] 1469:      pop ds
[XK4] 1470:      mov si,offset modusmeldung
[KS1] 1471:      call write
[WH3] 1472:      cmp es:[automodus],0
[YA2] 1473:      jne automatik
[AU3] 1474:      mov dl,es:[usermodus]
[ZC1] 1475:      push dx
[DA1] 1476:      mov cl,4
[XE1] 1477:      shr dl,cl
[792] 1478:      call writehex
[X31] 1479:      pop dx
[LAL] 1480:      and dl,15
[792] 1481:      call writehex
[UH4] 1482:      mov si,offset seitemeldung
[KS1] 1483:      call write
[TT3] 1484:      mov dl,es:[userseite]
[792] 1485:      call writehex
[RD2] 1486:      jmp short crlf
[CF2] 1487:      falschoption:
[LW4] 1488:      mov si,offset schalterfehler
[6Y2] 1489:      jmp fehlerausgabe
[UU1] 1490:      automatik:
[HB4] 1491:      mov si,offset automeldung
[KS1] 1492:      call write
[6V0] 1493:      crlf:
[DS3] 1494:      mov si,offset linefeed
[KS1] 1495:      call write
[6G3] 1496:      cmp es:[hercmodus],0

```

Listing 1. Der Quellcode des Programms »snapshot.asm«


```

[4U2] 1497: je keinhercgrafik
[8F4] 1498: mov si,offset hercmeldung1
[KS1] 1499: call write
[VR3] 1500: mov dl,es:[hercseite]
[792] 1501: call writehex
[BF4] 1502: mov si,offset hercmeldung2
[KS1] 1503: call write
[AR2] 1504: keinhercgrafik:
[CS4] 1505: ;Ist es ein Erstaufwurf?
[2H1] 1506: mov ax,cs
[6H1] 1507: mov bx,es
[TF1] 1508: cmp ax,bx
[142] 1509: jne zurueck
[SA5] 1510: ;Ja, dann resident installieren
[3D3] 1511: mov dx,offset pfad+64
[N81] 1512: int 27h
[9G1] 1513: zurueck:
[VM1] 1514: mov ah,4ch
[381] 1515: int 21h
[000] 1516:
[BU7] 1517: ;Den String [DS:SI] ausgeben, abschluss
mit 0
[BG1] 1518: writeit:
[MF1] 1519: mov dl,al
[XL1] 1520: mov ah,06h
[381] 1521: int 21h
[M21] 1522: write:
[CV0] 1523: lodsb
[BC1] 1524: and al,al
[342] 1525: jne writeit
[3P0] 1526: ret
[000] 1527:
[Y29] 1528: ;Das ASCII-Zeichen in AL auf gültige He
xziffer prüfen
[CT1] 1529: hexziffer:
[BE1] 1530: sub al,"0"
[EZ1] 1531: jc hexfehler
[MB1] 1532: cmp al,10
[DL1] 1533: jb hexende
[7C1] 1534: cmp al,48
[E82] 1535: jbe hexfehler
[8C1] 1536: sub al,39
[YB1] 1537: cmp al,15
[8Z1] 1538: ja hexfehler
[UD1] 1539: hexende:
[PL0] 1540: cnc
[3P0] 1541: ret
[ET1] 1542: hexfehler:
[EN0] 1543: stc
[3P0] 1544: ret
[000] 1545:
[PL5] 1546: ;Eine hexadezimale Ziffer ausgeben
[YK1] 1547: writehex:
[LA1] 1548: and dl,15
[2C1] 1549: add dl,"0"
[3F1] 1550: cmp dl,"9"
[EX2] 1551: jbe keinbuchstabe
[H61] 1552: add dl,7
[EE2] 1553: keinbuchstabe:
[NL1] 1554: mov ah,02h
[381] 1555: int 21h
[3P0] 1556: ret
[000] 1557:
[1Y6] 1558: programmname db 10,"SnapShot Version 2.5",0
[DH7] 1559: installmeldung db " installiert. Aufruf mit
"
[C12] 1560: if intr eq 5
[532] 1561: db "<PrtSc>",0
[2R0] 1562: else
[M45] 1563: db "<Alt-PrtSc> bzw. <SysReq>",0
[TU0] 1564: endif
[H17] 1565: laengefehler db 13,10,"Fehler: Der angegebene
"
[3Y5] 1566: db "Pfad ist länger als 62 Zeichen",0
[W09] 1567: schalterfehler db 13,10,"Fehler: Erlaubte O
ptionen sind "
[7V6] 1568: db "/Modus[,Seite], /A, /H[,Seite] und /M
",0
[M07] 1569: modusfehler db 13,10,"Fehler: Der angegeben
e "
[1L5] 1570: db "Videomodus ist nicht bekannt",0
[146] 1571: syntaxfehler db 13,10,"Fehler: Syntax "
[4L6] 1572: db "ist Snapshot [Pfad] [/Optionen]",0
[887] 1573: pfadmeldung db 13,10," Aktueller Dateiname
:",0
[HH7] 1574: modusmeldung db 13,10," Eingestellter Modu
s:",0
[L84] 1575: seitemeldung db ", Seite ",0
[FY6] 1576: automeldung db "Automatische Erkennung",0

```

```

[6X8] 1577: hercmeldung1 db " Im Modus 07 wird die Her
culesgrafik "
[MC2] 1578: db "auf Seite ",0
[7X4] 1579: hercmeldung2 db " gespeichert"
[RU2] 1580: linefeed db 13,10,0
[000] 1581:
[TF1] 1582: code ends
[SF1] 1583: end begin

```

Listing 1. Der Quellcode des Programms »snapshot.asm«

```

[PD5] 1: E9 5A 08 00 00 B8 08 D0 07 01 FF 01 01 00 B8 08
[ED5] 2: D0 07 01 FF 01 02 00 B8 10 A0 0F 01 FF 01 03 00
[KD5] 3: B8 10 A0 0F 01 FF 01 04 00 B8 40 40 3F 00 1A 02
[ZA5] 4: 05 00 B8 40 40 3F 00 1A 02 06 00 B8 40 40 3F 00
[TA5] 5: 36 02 07 00 B0 10 A0 0F 01 C4 02 0D 00 A0 20 40
[9D5] 6: 1F 00 74 03 0E 00 A0 40 80 3E 00 74 03 0F 00 A0
[385] 7: 80 60 6D 00 74 03 10 00 A0 80 60 6D 00 74 03 11
[B55] 8: 00 A0 00 00 96 00 14 02 12 00 A0 00 96 00 74
[WB5] 9: 03 13 00 A0 00 00 FA 00 2F 04 18 00 B0 40 60 2D
[L95] 10: 01 14 02 19 00 B0 10 C8 19 01 14 02 1A 00 B0 10
[K75] 11: E0 1C 01 14 02 22 00 B8 40 60 2D 01 14 02 23 00
[LB5] 12: B8 20 C8 19 01 14 02 24 00 B8 20 E0 1C 01 14 02
[Z75] 13: 25 00 A0 00 00 96 00 74 03 26 00 B8 40 80 25 01
[KA5] 14: 14 02 29 00 A0 00 60 EA 00 74 03 2A 00 B8 20 40
[UA5] 15: 1F 01 14 02 2D 00 A0 00 00 00 00 78 04 2E 00 A0
[M55] 16: 00 00 00 00 78 04 37 00 A0 00 00 00 00 74 03 E8
[9F5] 17: B3 03 BE 0B 02 E8 C8 03 E9 D3 03 07 00 B8 03 10
[RL5] 18: B3 01 CD 10 E8 9E 03 E9 C4 03 E8 98 03 BE 26 02
[EK5] 19: E8 AD 03 E9 B8 03 0E 00 B4 0B B8 10 00 CD 10 B4
[SN5] 20: 0B BB 00 01 CD 10 53 33 DB B4 06 CD 15 23 DB 5B
[UM5] 21: 74 D2 53 51 1E 8E 1E 7E 05 BA DE 03 33 F6 B9 A0
[VZ5] 22: 1F B0 00 EE 8B 1C FE C0 EE 39 1C 75 0F FE C0 EE
[PV5] 23: 39 1C 75 08 FE C0 EE AD 3B C3 E1 E5 1F 59 5B 74
[9R5] 24: A3 E8 41 03 BE A7 02 E8 56 03 BA DE 03 B0 00 EE
[1U5] 25: E8 5B 03 BE B5 02 E8 47 03 B0 01 EE E8 4F 03 BE
[NW5] 26: BA 02 E8 3B 03 B0 02 EE E8 43 03 BE BF 02 E8 2F
[CN5] 27: 03 B0 03 EE E9 37 03 0C 00 BA D9 03 B0 0F EE 83
[EF5] 28: C2 04 B0 01 EE 03 00 B0 02 EE 03 C6 B0 04 EE 03
[WD5] 29: 00 B0 08 EE 80 3E 8F 07 00 75 58 C0 06 00 07 00
[LK5] 30: 1E B8 40 00 8E D8 A0 65 00 1F A8 02 74 0B A8 80
[7S5] 31: 74 41 C6 06 90 07 01 EB 3A BA B9 03 EE 42 EC A8
[5R5] 32: 02 74 08 42 EE 4A EC A8 02 74 03 E9 16 FF EC A8
[5X5] 33: 80 74 FB EC A8 80 75 FB 42 EE 83 EA 02 EE BA B4
[5S5] 34: 03 B0 10 EE 42 EC 8A E0 4A B0 11 EE 42 EC 3D 8E
[5R5] 35: 0B 72 D8 C6 06 9C 05 00 E8 8A 02 BE 40 03 E8 9F
[6T5] 36: 02 A0 90 07 B4 08 F6 E4 8A F8 B9 EE 7E E9 9E 02
[KG5] 37: 32 00 BA BF 03 B0 03 EE E8 18 00 00 35 01 2D 02
[BB5] 38: 2E 03 07 04 5B 05 02 06 57 07 57 08 02 09 03 0A
[6W5] 39: 00 0B 00 5E B9 0C 00 BA B4 03 AD EF E2 FC BA B8
[NR5] 40: 03 B0 0A EE E8 3E 02 80 3E 79 05 0F 74 06 BE ED
[XP5] 41: 03 E8 4C 02 BE 04 E8 46 02 BA CE 03 B0 04 EE
[0M5] 42: 42 B0 00 EE E8 47 02 80 3E 79 05 0F 74 0C BE 19
[YG5] 43: 04 E8 2C 02 B0 01 EE E8 34 02 80 3E 79 05 29 74
[9H5] 44: 1F BE 1E 04 E8 19 02 B0 02 EE E8 21 02 80 3E 79
[AL5] 45: 05 0F 74 0C BE 23 04 E8 06 02 B0 03 EE E8 0E 02
[UC5] 46: 23 C9 75 18 80 3E 79 05 37 75 11 C7 06 7E 05 00
[GN5] 47: B0 B9 00 80 BE 28 04 E8 E6 01 EB 98 C3 1E 00 E8
[5Z4] 48: 11 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[7F5] 49: 00 00 00 5A 06 0E 07 B8 02 10 CD 10 07 0A 00 BA
[KG5] 50: C4 03 B0 02 EE 42 B0 01 EE 03 00 B0 02 EE 03 00
[EM5] 51: B0 04 EE 03 B0 B0 08 EE 05 00 B8 00 B0 8E C0 E8
[CF5] 52: 83 01 E8 03 00 E9 A6 01 BE 62 04 E8 92 01 53 51
[SK5] 53: 33 DB B8 15 10 CD 10 8A C6 E8 15 02 8A C5 E8 10
[WT5] 54: 02 8A C1 E8 0B 02 FE C3 75 E8 59 5B BE 67 04 E9
[BE5] 55: 6E 01 03 00 E8 FD 02 0F 00 5A 06 0E 07 B8 12 10
[3R5] 56: 33 DB B9 00 01 CD 10 07 E8 3A 01 E8 BA FF BE D5
[CL5] 57: 04 E8 4C 01 BA CD 03 B0 40 EE E8 51 01 BE DD 04
[2M5] 58: E8 3D 01 B0 48 EE E8 45 01 BE E2 04 E8 31 01 B0
[ZN5] 59: 50 EE E8 39 01 BE E7 04 E8 25 01 B0 58 EE 80 3E
[3M5] 60: 79 05 2D 75 06 B9 00 6B E9 23 01 E8 20 01 BE EC
[AG5] 61: 04 E8 0C 01 B0 60 EE 80 3E 79 05 2E 75 06 B9 00
[PK5] 62: B0 E9 0A 01 C3 06 00 BA CD 03 B0 40 EE 03 B0 B0
[LP5] 63: 41 EE 03 00 B0 42 EE 03 00 B0 43 EE 03 00 B0 44
[BD5] 64: EE 1D 00 4D 5A 00 00 00 00 00 00 20 00 00 00 FF
[X55] 65: FF 00 00 40 00 00 00 00 00 00 04 00 1E 00 00 01
[XN5] 66: 71 00 FC 1E 0E 1F EB 59 1F 81 3E 5D 00 43 20 74
[EG5] 67: 10 B9 00 10 B4 01 CD 10 B4 00 CD 16 B8 00 00 CD
[GH5] 68: 10 B4 4C CD 21 33 FF B5 00 81 FE 00 80 72 0B 81
[WS5] 69: EE 00 80 8C D8 05 D8 06 8E D8 AC 8A C8 80 F9 80
[VZ5] 70: 72 09 74 0B AC F5 D9 F3 AA BE DE F3 A4 EB DA 8B
[MR5] 71: FE B1 04 D3 EF 8C D8 03 C7 8E D8 50 83 E6 0F 56

```



```
[BG5] 72: CB B4 0F CD 10 A2 1B 00 B8 00 00 CD 10 B8 00 00
[CD5] 73: 8E C0 0E 10 00 E8 00 00 5E 81 C6 0D 00 58 50 50
[RE5] 74: B8 23 00 50 CB 05 00 B8 06 00 50 CB 00 54 05 58
[HF5] 75: 05 5C 05 F9 05 FE 05 00 06 17 06 2C 06 2D 06 3E
[EF5] 76: 06 00 00 00 00 BE 9D 05 57 AD 23 C0 74 0E 8B F8
[OT5] 77: 8A 05 24 FE 0A 06 9C 05 88 05 EB ED 5F BE 10 05
[6V5] 78: 51 AD 8B C8 E3 06 AC E8 87 00 E2 FA 59 C3 BE 83
[1U5] 79: 05 E8 EC FF 53 51 52 1E 8E 1E 7E 05 8B F3 03 CB
[KR5] 80: EB 04 3B F1 74 52 56 B2 00 AC 8B D8 EB 0D AC 93
[EY5] 81: 3A D8 74 1D FE C2 80 FA 7F 74 06 3B F1 75 EF FE
[EY5] 82: C2 5E 8A C2 E8 4A 00 AC E8 37 00 FE CA 75 F8 EB
[AV5] 83: D1 80 FA 00 75 EB B2 FE 3B F1 74 0C AC 3A C3 75
[YU5] 84: 07 FE CA 80 FA 81 75 F0 8A C2 E8 24 00 5E AC FE
[WS5] 85: C2 75 FB E8 0C 00 EB AA B0 80 E8 14 00 1F 5A 59
[3G5] 86: 5B C3 2E 80 3E 9C 05 00 74 07 50 E8 03 00 58 8A
[3M5] 87: C4 AA 81 FF 00 01 72 23 FF 74 1F 53 51 52 1E
[7T5] 88: 2E 8B 1E B3 05 8B CF 0E 1F 33 D2 B4 40 CD 21 1F
[4P5] 89: 5A 59 5B 72 07 3B C7 75 03 33 FF C3 8B 1E B3 05
[4W5] 90: B4 3E CD 21 B8 07 0E CD 10 8B E5 C3 8B EC A0 80
[PK5] 91: 07 8A 3E 8D 07 80 3E 8E 07 00 74 04 B4 0F CD 10
[5P5] 92: BE 03 01 3A 04 74 0B 83 C6 09 81 FE FF 01 72 F3
[HT5] 93: EB D2 A2 79 05 B1 20 BA 53 09 B4 3C CD 21 A3 B3
[WU5] 94: 05 73 02 EB BF 33 FF 56 BE F1 04 E8 F2 FE 5E B9
[PV5] 95: 23 02 B0 08 E8 7A FF E2 F9 AC AD A3 7E 05 AC F6
[G5Y] 96: E7 B3 00 8A F8 AD BB C8 AC A2 9C 05 AD FF D0 BE
[FU5] 97: 95 05 E8 CB FE E8 60 FF B8 02 42 8B 1E B3 05 33
[9M5] 98: C9 33 D2 CD 21 72 BC 05 FF 01 83 D2 00 B9 00 02
[UR5] 99: F7 F1 AB B8 00 42 33 C9 BA 04 00 CD 21 72 A4 E8
[OW5] 100: 36 FF 8B 1E B3 05 B4 3E CD 21 73 03 E9 55 FF BE
[7V5] 101: 53 09 AC 3C 00 75 FB FD AC 81 FE 53 09 72 1C AC
[RR5] 102: 3C 5C 74 17 3C 30 72 F1 3C 39 77 ED FE C0 88 44
[9D5] 103: 01 3C 39 76 06 C6 44 01 30 EB DE C3 93 07 00 00
[975] 104: 22 08 00 00 C2 08 00 00 BB 07 00 00 C7 08 00 00
[X65] 105: DB 08 00 00 05 09 00 00 00 00 00 00 00 FF 00
[5H5] 106: 00 00 00 50 1E 33 C0 8E D8 8C C8 39 06 26 00 74
[VE5] 107: 13 87 06 26 00 2E A3 7A 07 B8 BB 07 87 06 24 00
[LM5] 108: 2E A3 78 07 1F 58 2E FF 2E 6C 07 50 E4 60 3C 38
[FN5] 109: 75 08 2E C6 06 91 07 FF EB 41 3C B8 75 08 2E C6
[FD5] 110: 06 91 07 00 EB 35 3C 54 74 0C 3C 37 75 2D 2E 80
[YC5] 111: 3E 91 07 00 74 25 2E 80 3E 92 07 00 75 0D 2E C6
[DN5] 112: 06 92 07 FF 56 57 E8 29 01 5F 5E E4 61 0C 80 E6
[MM5] 113: 61 24 7F E6 61 B0 61 E6 20 58 CF 58 2E FF 2E 78
[LC5] 114: 07 00 01 02 03 04 05 14 07 38 39 3A 3B 3C 3D 3E
[PL5] 115: 3F 00 22 E4 74 0F 80 FC 0B 74 33 80 FC 10 73 41
[CG5] 116: 2E FF 2E 70 07 2E C6 06 11 02 01 2E C6 06 2B 02
[5B5] 117: 10 2E C6 06 32 02 00 50 53 BB 10 00 2E 8A 87 11
[G5V] 118: 08 2E 88 87 F2 03 FE CB 59 F2 5B 58 EB D2 80 FF
[CL5] 119: 00 75 07 2E 88 1E 2B 02 EB C6 2E 88 1E 32 02 EB
[HM5] 120: BF 3D 00 10 75 14 80 F9 10 73 B5 57 8B F9 81 E7
[WM5] 121: FF 00 2E 88 BD F2 03 5F EB A6 3D 01 10 75 07 2E
[FG5] 122: 88 3E 02 04 EB 9A 3D 02 10 75 1A 50 53 57 8B FA
[3S5] 123: BB 10 00 26 8A 01 2E 88 87 F2 03 FE CB 79 F4 5F
[LN5] 124: 5E 58 E9 7B FF 3D 03 10 75 05 2E 88 1E 11 02 E9
[3U5] 125: 6E FF E8 43 00 CD 19 9C 2E FF 1E 7C 07 1E 56 2E
[GE5] 126: C5 36 88 07 80 3C 00 5E 1F 75 29 50 53 51 52 56
[1T5] 127: 57 55 1E 06 E8 3C 00 FB E8 4E 00 8C 8E C0 E8
[8S5] 128: AA FD E8 44 00 07 1F 5D 5F 5E 5A 59 5B 58 2E C6
[FS5] 129: 06 92 07 00 CF B0 03 CF BF 20 00 BE 6C 07 E8 2E
[9G5] 130: 00 BF 40 00 BE 70 07 E8 25 00 BF 64 00 BE 74 07
[AP5] 131: EB 1D 1E 06 BF 20 00 BE 7C 07 E8 12 00 BF A0 00
[7P5] 132: BE 80 07 E8 09 00 07 1F C3 BF 90 00 BE 84 07 FC
[2N5] 133: 8C C8 8E D8 33 C0 8E C0 E8 00 26 80 05 A5 89
[1H5] 134: 44 FE C3 42 49 4C 44 30 2E 45 58 45 00 BE 33 0B
[MR5] 135: E8 A1 01 BF 6E 07 8C C8 B9 07 00 AB 83 C7 02 E2
[VV5] 136: FA B8 09 35 CD 21 BE BB 07 8B FB B9 20 00 F3 A7
[SS5] 137: 74 18 BE 49 0B E8 7C 01 FA E8 7C FF B4 34 CD 21
[1L5] 138: 89 1E 88 07 8C 06 8A 07 0E 07 BE 80 00 AC B4 00
[ME5] 139: 03 F0 C6 04 00 BE 81 00 AC 3C 00 74 17 3C 20 74
[2S5] 140: F7 3C 2F 74 30 BF 53 09 B4 40 FE CC 75 09 BE 7C
[OM5] 141: 0B E8 40 01 E9 B6 00 3C 00 74 F9 3C 2F 74 16 3C
[1K5] 142: 20 74 1F 3C 61 72 06 3C 7A 77 02 24 DF AA 26 C6
[7M5] 143: 05 00 AC EB D5 AC 0C 20 3C 61 75 09 26 C6 06 8E
[US5] 144: 07 FF EB 76 90 3C 68 75 21 26 C6 06 8F 07 FF AC
[LH5] 145: 3C 2C 75 67 AC 3C 30 74 07 3C 31 74 03 E9 B0 00
[AN5] 146: 2C 30 26 A2 90 07 EB 52 EB CB 3C 6D 75 08 26 C6
[7G5] 147: 06 8F 07 00 EB 44 E8 E1 00 72 39 B1 04 D2 E0 50
[3P5] 148: AC 0C 20 E8 D4 00 59 72 2B 0A C1 BB 03 01 26 3A
[RT5] 149: 07 74 0E 83 C3 09 81 FB FF 01 72 F2 BE FB 0B EB
[GR5] 150: 29 26 A2 8C 07 26 C6 06 8E 07 00 AC 3C 2C 75 0B
[DN5] 151: AC E8 A6 00 72 5A 26 A2 8D 07 AC 3C 00 74 0E 3C
[VT5] 152: 20 74 F7 3C 2F 74 A1 BE 31 0C E8 87 00 FB BE 62
[CR5] 153: 0C E8 80 00 1E 06 1F BE 53 09 E8 77 00 1F BE 7C
[LG5] 154: 0C E8 70 00 26 80 3E 8E 07 00 75 29 26 8A 16 8C
[UK5] 155: 07 52 B1 04 D2 EA E8 77 00 5A 80 E2 0F E8 70 00
[5R5] 156: BE 96 0C E8 4E 00 26 8A 16 8D 07 E8 62 00 EB 0B
[SW5] 157: BE B4 0B EB B5 BE 9F 0C E8 39 00 BE F3 0C E8 33
[OK5] 158: 00 26 80 3E 8F 07 00 74 14 BE B6 0C E8 25 00 26
[RS5] 159: 8A 16 90 07 E8 39 00 BE E7 0C E8 17 00 8C C8 8C
[5T5] 160: C3 3B C3 75 05 BA 93 09 CD 27 B4 4C CD 21 8A D0
```

```
[CK5] 161: B4 06 CD 21 AC 22 C0 75 F5 C3 2C 30 72 10 3C 0A
[4K5] 162: 72 0A 3C 30 76 08 2C 27 3C 0F 77 02 F8 C3 F9 C3
[9F5] 163: 80 E2 0F 80 C2 30 80 FA 39 76 03 80 C2 07 B4 02
[NG5] 164: CD 21 C3 0A 53 6E 61 70 53 68 6F 74 20 56 65 72
[EG5] 165: 73 69 6F 6E 20 32 2E 35 00 20 69 6E 73 74 61 6C
[DE5] 166: 6C 69 65 72 74 2E 20 41 75 66 72 75 66 20 6D 69
[WD5] 167: 74 20 3C 41 6C 74 2D 50 72 74 53 63 3E 20 62 7A
[DE5] 168: 77 2E 20 3C 53 79 73 52 65 71 3E 00 0D 0A 46 65
[AF5] 169: 68 6C 65 72 3A 20 44 65 72 20 61 6E 67 65 67 65
[170] 170: 62 65 6E 65 20 50 66 61 64 20 69 73 74 20 6C 84
[4C5] 171: 6E 67 65 72 20 61 6C 73 20 36 32 20 5A 65 69 63
[KE5] 172: 68 65 6E 00 0D 0A 46 65 68 6C 65 72 3A 20 45 72
[MH5] 173: 6C 61 75 62 74 65 20 4F 70 74 69 6F 6E 65 6E 20
[SK5] 174: 73 69 6E 64 20 2F 4D 6F 64 75 73 5B 2C 53 65 69
[NK5] 175: 74 65 5D 2C 20 2F 41 2C 20 2F 48 5B 2C 53 65 69
[VF5] 176: 74 65 5D 20 75 6E 64 20 2F 4D 6F 64 75 73 5B 2C 53 65 69
[RD5] 177: 6C 65 72 3A 20 44 65 72 20 61 6E 67 65 67 65 62
[SF5] 178: 65 6E 65 20 56 69 64 65 6F 6D 6F 64 75 73 20 69
[2G5] 179: 73 74 20 6E 69 63 68 74 20 62 65 6B 61 6E 6E 74
[AD5] 180: 00 0D 0A 46 65 68 6C 65 72 3A 20 53 79 6E 74 61
[6G5] 181: 78 20 69 73 74 20 53 6E 61 70 73 68 6F 74 20 5B
[BN5] 182: 58 65 61 64 5D 20 5B 2F 4F 70 74 69 6F 6E 65 6E
[EG5] 183: 5D 00 0D 0A 20 20 41 6B 74 75 65 6C 6C 65 72 20
[8E5] 184: 44 61 74 65 69 6E 61 6D 65 3A 20 00 0D 0A 20 20
[LH5] 185: 45 69 6E 67 65 73 74 65 6C 6C 74 65 72 20 4D 6F
[G95] 186: 64 75 73 3A 20 00 2C 20 53 65 69 74 65 20 00 41
[WD5] 187: 75 74 6F 6D 61 74 69 73 63 68 65 20 45 72 6B 65
[4L5] 188: 6E 6E 75 6E 67 00 20 20 49 6D 20 4D 6F 64 75 73
[BN5] 189: 20 30 37 20 77 69 72 64 20 64 69 65 20 48 65 72
[OB5] 190: 63 75 6C 65 73 67 72 61 66 69 6B 20 61 75 66 20
[HA5] 191: 53 65 69 74 65 20 00 00 67 65 73 70 65 69 63 68
[9Y1] 192: 65 72 74 0D 0A 00
```

Listing 2. Der Hexdump »snapshot.com«

```
[AS4] 1: {SR-,S-,I-,D-,F-,V-,B-,M 8192,0,0}
[AS0] 2: Uses
[9G1] 3: Crt, Dos;
[000] 4:
[RN0] 5: Var
[XD3] 6: GrafikFlag: Boolean;
[Z53] 7: Datei: File Of Byte;
[LW2] 8: Bildname: String;
[NL7] 9: VideoModus, Wert, Hintergrund, Palette: B
yte;
[3S2] 10: Reg: Registers;
[WY1] 11: Taste: Char;
[SH6] 12: Rot, Gruen, Blau: Array [0..255] Of Byte;
[U04] 13: Farben, Max, I: Integer;
[983] 14: Blinkflag: Boolean;
[000] 15:
[XV5] 16: Procedure Fehler (Meldung: String);
[MU0] 17: Begin
[KF6] 18: If GrafikFlag Then TextMode (LastMode);
[AS4] 19: Writeln ('Fehler: ', Meldung);
[W31] 20: Halt (1)
[ZN0] 21: End;
[000] 22:
[PP7] 23: Procedure SetzePosition (Position: LongInt)
;
[MU0] 24: Begin
[7V3] 25: Seek (Datei, Position);
[E7A] 26: If IOResult <> 0 Then Fehler ('Datei ka
nn nicht gelesen werden')
[ZN0] 27: End;
[000] 28:
[6H3] 29: Procedure ZeigeBild;
[MU0] 30: Begin
[U23] 31: Exec (Bildname, 'C');
[BHA] 32: If DosExitCode <> 6 Then Fehler ('Bild
kann nicht geladen werden');
[R43] 33: GrafikFlag := True;
[753] 34: SetzePosition (692)
[ZN0] 35: End;
[000] 36:
[1C2] 37: Procedure CGA;
[MU0] 38: Begin
[7R1] 39: ZeigeBild;
[X94] 40: Read (Datei, Hintergrund);
[R73] 41: SetzePosition (699);
[RH3] 42: Read (Datei, Palette);
[PH8] 43: Palette := (Palette And 1) Or (Hintergr
und Shr 3 And 2);
```

Listing 3. Das Programm »palette.pas« zum Ändern der Farben


```

[P31] 44: Repeat
[XT9] 45:   Hintergrund := (Hintergrund And 15) Or
      r (Palette Shl 3 And 16);
[ØX1] 46:   Reg.AH := $0B;
[3N3] 47:   Reg.BX := Hintergrund;
[B42] 48:   Intr ($10, Reg);
[ØX1] 49:   Reg.AH := $0B;
[2R4] 50:   Reg.BX := 256 Or (Palette And 1);
[B42] 51:   Intr ($10, Reg);
[HR2] 52:   Taste := ReadKey;
[Z12] 53:   Case Taste Of
[6P3] 54:     'h': Inc (Hintergrund);
[LM3] 55:     'H': Dec (Hintergrund);
[ZV2] 56:     'p': Inc (Palette);
[CK2] 57:     'P': Dec (Palette);
[ZØØ] 58:     #13:
[ØØØ] 59:       Begin
[273] 60:         SetzePosition (692);
[TP4] 61:         Write (Datei, Hintergrund);
[R73] 62:         SetzePosition (699);
[ØU3] 63:         Palette := Palette And 1;
[EG3] 64:         Write (Datei, Palette)
[4MØ] 65:       End
[4MØ] 66:     End
[PS3] 67:   Until Taste In [#13, #27]
[ZNØ] 68: End;
[ØØØ] 69:
[ØØØ] 70: Procedure HoleTaste;
[ØØØ] 71: Begin
[HR2] 72:   Taste := ReadKey;
[LS4] 73:   Blinkflag := Taste In [#Ø, ' '];
[EN2] 74:   If Taste = #Ø Then
[ØM2] 75:     Case ReadKey Of
[YP1] 76:       #54D: Inc (I);
[ZN1] 77:       #54B: Dec (I);
[XX1] 78:       #574: I := I + 8;
[ZV1] 79:       #573: I := I - 8;
[2H1] 80:       #547: I := Ø;
[ZV1] 81:       #54F: I := Max
[ZNØ] 82:     End;
[ØØØ] 83:   If I > Max Then I := Max;
[3S2] 84:   If I < Ø Then I := Ø;
[ØT2] 85:   If Taste = 'Ø' Then
[ØØØ] 86:     Begin
[R32] 87:       Rot [I] := Ø;
[ØØØ] 88:       Gruen [I] := Ø;
[NZ1] 89:       Blau [I] := Ø
[ZNØ] 90:     End;
[5EB] 91:   If (Taste In ['r', 'm', 'y', 'w']) And
      (Rot [I] < Farben) Then Inc (Rot [I]);
[V8A] 92:   If (Taste In ['R', 'M', 'Y', 'W']) And
      (Rot [I] > Ø) Then Dec (Rot [I]);
[G2C] 93:   If (Taste In ['g', 'y', 'c', 'w']) And
      (Gruen [I] < Farben) Then Inc (Gruen [I]);
[PXA] 94:   If (Taste In ['G', 'Y', 'C', 'W']) And
      (Gruen [I] > Ø) Then Dec (Gruen [I]);
[4FB] 95:   If (Taste In ['b', 'm', 'c', 'w']) And
      (Blau [I] < Farben) Then Inc (Blau [I]);
[Ø6A] 96:   If (Taste In ['B', 'C', 'M', 'W']) And
      (Blau [I] > Ø) Then Dec (Blau [I])
[ZNØ] 97:   End;
[ØØØ] 98:
[9C2] 99: Procedure EGA;
[GL3] 100:   Function Farbe: Byte;
[ØØØ] 101:   Begin
[ØØØ] 102:     Farbe := (Rot [I] Shl 5 And 32) Or (R
      ot [I] Shl 1 And 4)
[LA7] 103:     Or (Gruen [I] Shl 4 And 16) Or (Gru
      en [I] And 2)
[ZB7] 104:     Or (Blau [I] Shl 3 And 8) Or (Blau
      [I] Shr 1 And 1)
[ZNØ] 105:   End;
[ØØØ] 106:   Begin
[7R1] 107:     ZeigeBild;
[WP2] 108:     For I := Ø To 16 Do
[ØØØ] 109:       Begin
[523] 110:         Read (Datei, Wert);
[DR7] 111:         Rot [I] := (Wert Shr 5 And 1) Or (W
      ert Shr 1 And 2);
[K57] 112:         Gruen [I] := (Wert Shr 4 And 1) Or
      (Wert And 2);
[AP7] 113:         Blau [I] := (Wert Shr 3 And 1) Or (
      Wert Shl 1 And 2)
[ZNØ] 114:       End;
[8R1] 115:       Farben := 3;
[PE1] 116:       Max := 16;
[LYØ] 117:       I := Ø;
[P31] 118:     Repeat

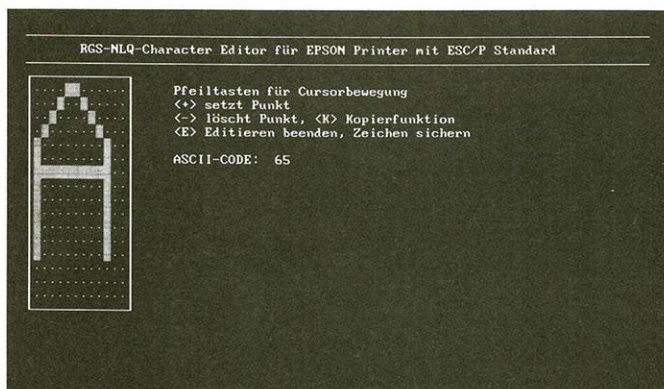
```

```

[EX6] 119:     If I < 16 Then Reg.AX := $1000 Else R
      eg.AX := $1001;
[SS1] 120:     Reg.BL := I;
[ZD2] 121:     Reg.BH := Farbe;
[B42] 122:     Intr ($10, Reg);
[ST1] 123:     HoleTaste;
[416] 124:     If Blinkflag Then For Wert := Ø To 3
      Do
[ØØØ] 125:       Begin
[EX6] 126:         If I < 16 Then Reg.AX := $1000 El
      se Reg.AX := $1001;
[SS1] 127:         Reg.BL := I;
[ØX3] 128:         Reg.BH := (Wert And 1) * 63;
[G22] 129:         Intr ($10, Reg)
[4MØ] 130:       End
[7Y3] 131:     Until Taste In [#13, #27];
[ØU2] 132:     If Taste = #13 Then
[ØØØ] 133:       Begin
[273] 134:         SetzePosition (692);
[WP2] 135:         For I := Ø To 16 Do
[ØØØ] 136:           Begin
[P92] 137:             Wert := Farbe;
[Z33] 138:             Write (Datei, Wert)
[4MØ] 139:           End
[4MØ] 140:         End
[ZNØ] 141:       End;
[ØØØ] 142:
[ØØØ] 143: Procedure VGA;
[ØØØ] 144: Begin
[7R1] 145:   ZeigeBild;
[AE9] 146:   For I := Ø To 255 Do Read (Datei, Rot [
      I], Gruen [I], Blau [I]);
      Farben := 63;
      Max := 255;
      I := Ø;
      Repeat
[2V1] 147:     Reg.AX := $1010;
[SS1] 152:     Reg.BL := I;
[GV2] 153:     Reg.DH := Rot [I];
[E93] 154:     Reg.CH := Gruen [I];
[ØØ3] 155:     Reg.CL := Blau [I];
[B42] 156:     Intr ($10, Reg);
[ST1] 157:     HoleTaste;
[XZ5] 158:     If Blinkflag Then For Wert := Ø To 31
      Do
[ØØØ] 159:       Begin
[A42] 160:         Reg.AX := $1010;
[WT1] 161:         Reg.BX := I;
[4X3] 162:         Reg.DH := (Wert And 1) * 63;
[2X3] 163:         Reg.CH := (Wert And 1) * 63;
[EX3] 164:         Reg.CL := (Wert And 1) * 63;
[B42] 165:         Intr ($10, Reg);
[X71] 166:         Delay (1)
[4MØ] 167:       End
[7Y3] 168:     Until Taste In [#13, #27];
[ØU2] 169:     If Taste = #13 Then
[ØØØ] 170:       Begin
[273] 171:         SetzePosition (692);
[PT9] 172:         For I := Ø To 255 Do Write (Datei,
      Rot [I], Gruen [I], Blau [I])
[4MØ] 173:       End
[ZNØ] 174:     End;
[ØØØ] 175:
[ØØØ] 176: Begin
[963] 177:   GrafikFlag := False;
[DVA] 178:   If ParamCount <> 1 Then Fehler ('Syntax i
      st "Palette <Bildname>"');
      If Pos ('.', ParamStr (1)) > Ø Then
[XT4] 179:     Bildname := ParamStr (1)
[LV3] 180:   Else
[ZRØ] 181:     Bildname := ParamStr (1) + '.EXE';
[A55] 182:   Assign (Datei, Bildname);
[SY3] 183:   Reset (Datei);
[E62] 184:   If IOResult <> Ø Then Fehler ('Datei nich
      t gefunden');
[L73] 186:   SetzePosition (679);
[154] 187:   Read (Datei, VideoModus);
[173] 188:   Case VideoModus Of
[F11] 189:     $Ø4, $Ø5: CGA;
[7B3] 190:     $ØD, $ØE, $10, $12, $37: EGA;
[GA2] 191:     $13, $2D, $2E: VGA;
[CA6] 192:     Else Fehler ('Unbekanntes Bildformat')
[ZNØ] 193:   End;
[552] 194:   Close (Datei);
[BC3] 195:   TextMode (LastMode);
[VH8] 196:   If Taste = #27 Then Writeln ('Abbruch dur
      ch Benutzer')
[KNØ] 197: End.

```

Listing 3. Das Programm »palette.pas« zum Ändern der Farben



Die Oberfläche des Programms »cedit«

Druckerzeichensatz selbst gemacht

Zählen Sie auch zu den Druckerbenutzern, die das Schriftbild Ihres Epson-Kompatiblen langweilt; die sich wünschen, einmal andere Schriftarten zu sehen? Mit den Programmen »prnlq.bas« und »cedit.bas« fertigen Sie sich Ihre eigenen Zeichensätze an und drucken damit beliebige Texte.

Der IBM-Zeichensatz Epson-kompatibler Drucker trägt seinen Namen nicht ganz zurecht, er ist nämlich genau betrachtet keiner. Wohl ist ein Großteil der Zeichen mit dem IBM-Zeichensatz identisch, doch sobald ein Sonderzeichen den Weg durch das Parallelkabel antritt, zeigt der Drucker ungerührt ein ganz anderes Zeichen. Mit dem Programm »cedit« (Listing 1), das für Turbo-Basic geschrieben ist, kann man beliebige Zeichensätze anfertigen und mit »prnlq« (Listing 2) reine Textdateien in diesem Zeichensatz ausdrucken. Zwar sind die Programme in Turbo-Basic programmiert, doch sollte eine Konvertierung auf einen anderen Basicdialekt nicht schwerfallen. Beide Programme sind von erfrischender Einfachheit und erklären sich selbst.

Tippen Sie die Programme mit SCheck aus dieser Ausgabe ab und compilieren Sie sie mit Turbo-Basic. Listing 3, 4 und 5 sind drei Beispiel-Zeichensatzdateien, wobei »standard.tbl« ein tatsächlicher IBM-Zeichensatz ist. Einen Ausdruck nehmen Sie vor, indem Sie das Programm »prnlq« starten, den Namen der Zeichensatzdatei ohne Erweiterung eingeben und dem Programm den Namen der Textdatei bekanntgeben. Diese druckt Ihr Epson-Kompatibler dann im Grafikmodus, es geht also etwas langsamer, als Sie es sonst von ihm gewohnt sind. Das Programm nimmt den Drucker an der parallelen Schnittstelle »LPT1« an, sollte Ihr Drucker an einer anderen parallelen Schnittstelle angeschlossen sein, ändern Sie im Programm bitte alle Vorkommen des Gerätenamens um.

Um eigene Zeichensätze anzufertigen, empfiehlt es sich, einen bereits vorhandenen Zeichensatz (eines der Listings 3, 4 oder 5) auf eine Datei des gewünschten Namens zu kopieren, so zum Beispiel

```
copy standard.tbl meinsatz.tbl
```

Rufen Sie das Programm »cedit« auf (Bild) und wählen Sie die Modifikation eines bereits vorhandenen Zeichensatzes an. Nachdem das Programm den Dateinamen des Zeichensatzes (in diesem Beispiel »meinsatz«) ohne Erweiterung erfuhr, wählen Sie ein Zeichen aus, indem Sie seinen ASCII-Code angeben oder es mit den Cursor-Tasten aus-

chen. Ändern Sie das Zeichen nach Belieben durch Setzen und Löschen der einzelnen Punkte. Auf Wunsch kopiert das Programm einzelne Zeichen an die Position anderer. Speichern Sie die Zeichensatz-Datei, und drucken Sie Ihren Text wie oben beschrieben.

(Roland Graf/db)

```
[3DB] 1: '** RGS - Roland Graf's Software - powerful
      computing - *****
[M96] 2: '*'
[3S8] 3: '* Programmname.....: PRNLQ.BAS
      *
[X1C] 4: '* Aufgabe.....: Lesen einer mit CE
      DIT erzeugten Zeichensatztabelle
[SRA] 5: '* und Ausgabe im Gra
      fikmodus in NLQ auf EPSON kompa-
[LV7] 6: '* tiblem Drucker
      *
[VZ8] 7: '* Programmierer.....: Roland Graf
      *
[UM7] 8: '* Version.....: 1.2
      *
[598] 9: '* Erstellt am.....: 01.02.1988
      *
[4B8] 10: '* Letzter Update.....: 08.02.1988
      *
[X89] 11: '* Programmiersprache..: Turbo BASIC
      *
[M96] 12: '*'
      *
[H9D] 13: '* (c)1988 Redaktionsbüro Everts und Hagedo
      rn
      *
[058] 14: '*****
      *****
[000] 15:
[1M1] 16: ProgInit:
[ND3] 17: ON ERROR GOTO Fehler3
[EC2] 18: WIDTH "LPT1:",970
[KY8] 19: DIM C(255,2,12) '256 ASCII Codes, 2-
      geteilt, 12-Dez.-Zahlen
[HD3] 20: ON ERROR GOTO Fehler1
[000] 21:
[000] 22:
[631] 23: NLQPR:
[PN0] 24: CLS
[0V6] 25: ? "RGS-NLQ-Printer DOS International 198
      8"
[9F1] 26: LOCATE 5,1
[BUB] 27: INPUT "Bitte geben Sie den Namen des gewü
      nschten Zeichensatzes ein: ",NA$
[VD4] 28: IF NA$="" THEN GOTO DirZeich
[CL3] 29: IF NA$="" THEN GOTO Ende
[867] 30: IF LEN(NA$)>8 OR INSTR(NA$,".") THEN GOTO
      ZuLang:
[4X1] 31: NA$=NA$+".TBL"
[FE3] 32: LOCATE 5,1: ? SPACE$(80)
[FN7] 33: LOCATE 5,1: ? "Zeichensatz ";NA$;" wird g
      eladen..."
[W32] 34: OPEN "I",#1,NA$
[5B2] 35: FOR A%=0 TO 255
[FM8] 36: INPUT#1,C(A%,1,1),C(A%,1,2),C(A%,1,3),C(A
      %,1,4),C(A%,1,5),C(A%,1,6),
[7D7] 37: C(A%,1,7),C(A%,1,8),C(A%,1,9),C(A%,1,10),
      C(A%,1,11),C(A%,1,12)
[VM8] 38: INPUT#1,C(A%,2,1),C(A%,2,2),C(A%,2,3),C(A
      %,2,4),C(A%,2,5),C(A%,2,6),
[PD7] 39: C(A%,2,7),C(A%,2,8),C(A%,2,9),C(A%,2,10),
      C(A%,2,11),C(A%,2,12)
[P51] 40: NEXT A%
[G51] 41: CLOSE#1
[SK8] 42: LOCATE 5,1: ? SPACE$(80);: LOCATE 5,1:?"Z
      eichensatz: ";NA$
[000] 43:
[KD1] 44: GetName:
[LD3] 45: ON ERROR GOTO Fehler2
[127] 46: LOCATE 7,1: ? SPACE$(80): LOCATE 8,1: ? S
      PACE$(80)
[FE9] 47: LOCATE 7,1: INPUT "Name der zu druckenden
      ASCII-Datei: ",DR$
[4M3] 48: IF DR$="" THEN GOTO Ende
[V42] 49: OPEN "I",#1,DR$
[FM0] 50: EK=0
[000] 51:
[3E1] 52: GetLine:
[ND3] 53: ON ERROR GOTO Fehler3
[EP2] 54: LINE INPUT#1,TX$
[YR2] 55: IF EOF(1) THEN EK=1
[8N3] 56: 'String in NLQ drucken
```

Listing 1. Einen eigenen Zeichensatz anfertigen mit »cedit.bas«


```

[WW1] 57: Z=LEN(TX$)*12
[4S1] 58: n1=Z mod 256
[HU1] 59: n2=INT(Z/256)
[UL6] 60: LPRINT CHR$(27);"***";CHR$(1);CHR$(n1);CHR$(
      (n2));
[X63] 61: FOR P%=1 TO LEN(TX$)
[213] 62: CD=ASC(MID$(TX$,P%,1))
[P91] 63: '1. Zeile
[Y92] 64: FOR PP%=1 TO 12
[XH3] 65: LPRINT CHR$(C(CD,1,PP%));
[ZS0] 66: NEXT
[ZS0] 67: NEXT
[ZG4] 68: 'Mikrovorschub und 2. Zeile
[6Y3] 69: LPRINT CHR$(27);"3";CHR$(1);
[H51] 70: LPRINT
[UL6] 71: LPRINT CHR$(27);"***";CHR$(1);CHR$(n1);CHR$(
      (n2));
[X63] 72: FOR P%=1 TO LEN(TX$)
[213] 73: CD=ASC(MID$(TX$,P%,1))
[T91] 74: '2. Zeile
[Y92] 75: FOR PP%=1 TO 12
[OK3] 76: LPRINT CHR$(C(CD,2,PP%));
[ZS0] 77: NEXT
[ZS0] 78: NEXT
[RT2] 79: LPRINT CHR$(27);"2";
[KN6] 80: LPRINT: IF EK=1 then CLOSE#1: GOTO GetNam
      e
[742] 81: GOTO GetLine
[000] 82:
[5A1] 83: Fehler1:
[F71] 84: COLOR 15
[Z69] 85: LOCATE 20,1:"Zeichensatzdatei existiert
      nicht!!! <TASTE>"
[WF2] 86: GOSUB ErrSound
[R22] 87: GOSUB GetKey
[K61] 88: COLOR 7
[3C3] 89: LOCATE 20,1: ? SPACE$(80)
[832] 90: RESUME NLQPR
[000] 91:
[9A1] 92: Fehler2:
[F71] 93: COLOR 15
[G48] 94: LOCATE 20,1:"Textdatei existiert nicht!!
      ! <TASTE>"
[WF2] 95: GOSUB ErrSound
[R22] 96: GOSUB GetKey
[K61] 97: COLOR 7
[3C3] 98: LOCATE 20,1: ? SPACE$(80)
[CD2] 99: RESUME GetName
[000] 100:
[DA1] 101: Fehler3:
[F71] 102: COLOR 15
[E88] 103: LOCATE 20,1: "Schalten Sie bitte den Dru
      cker ONLINE"
[RV7] 104: ? "oder drücken Sie eine Tast
      e um abzubreaken !"
[WF2] 105: GOSUB ErrSound
[3B4] 106: IF inkey$<">" THEN GOTO Ende
[X31] 107: RESUME
[000] 108:
[XB1] 109: ZuLang:
[F71] 110: COLOR 15
[CTB] 111: LOCATE 20,1:"Der von Ihnen eingegebene D
      ateiname ist ungültig! <TASTE>"
[WF2] 112: GOSUB ErrSound
[R22] 113: GOSUB GetKey
[K61] 114: COLOR 7
[3C3] 115: LOCATE 20,1: ? SPACE$(80)
[VU1] 116: GOTO NLQPR
[000] 117:
[NB1] 118: GetKey:
[UV5] 119: KE$=INKEY$: IF KE$="" THEN GOTO GetKey
[651] 120: RETURN
[000] 121:
[KH1] 122: DirZeich:
[PN0] 123: CLS
[EE6] 124: ? "Verzeichnis der Zeichensatzdateien:"
[Y30] 125: ?
[4W1] 126: FILES "*.TBL"
[Y30] 127: ?
[TL1] 128: ? "<TASTE>"
[R22] 129: GOSUB GetKey
[VU1] 130: GOTO NLQPR
[000] 131:
[8N1] 132: ErrSound:
[DP1] 133: SOUND 500,2
[TP1] 134: SOUND 800,2
[9P1] 135: SOUND 400,2
[5P1] 136: SOUND 300,2
[651] 137: RETURN
[000] 138:
[XT0] 139: Ende:
[PN0] 140: CLS
[U73] 141: ? "RGS-NLQP beendet."
[4M0] 142: END

```

Listing 1. Einen eigenen Zeichensatz anfertigen mit
»cedit.bas«

```

[CXA] 10 'Character Table Editor 16x12 Matrix für EPSON
      -NLQ-Simulation
[VC0] 20 '
[FG4] 30 'Programmname: CEDIT.BAS
[F04] 40 'Autor : Roland Graf
[9P3] 50 'Erstellt am : 05.01.88
[TP3] 60 'Last Update : 08.02.88
[5D0] 70 '
[2R7] 80 '(c) 1988 Redaktionsbüro Everts und Hagedorn
[9D0] 90 '
[FG4] 100 '***** Variablendefinition
[NN8] 110 DIM C(255,2,12) '256 ASCII-Codes, 2geteilt, 1
      2 Dez-Zahlen
[6H0] 120 '
[6M3] 130 UP$=CHR$(0)+CHR$(&H48)
[KK3] 140 DN$=CHR$(0)+CHR$(&H50)
[9M3] 150 RI$=CHR$(0)+CHR$(&H4D)
[GL3] 160 LE$=CHR$(0)+CHR$(&H4B)
[T61] 170 CH=65
[AXB] 180 DEF SEG=&HB800 'Wenn Sie eine Herculeskarte o
      der eine reine Textkarte
[B39] 185 'besitzen, setzen Sie hier B00
      0 statt B800 ein !!
[R12] 190 '***** Menue
[BG2] 200 KEY OFF: CLS
[XU3] 210 PRINT STRING$(80,196);
[NNC] 220 PRINT " RGS-NLQ-Character Editor für EP
      SON Printer mit ESC/P Standard"
[3V3] 230 PRINT STRING$(80,196);
[2U1] 240 GOSUB 420
[2F9] 250 LOCATE 6,20:PRINT"<L>aden Zeichentabelle und
      editieren"
[LP9] 260 LOCATE 8,20:PRINT"<N>eue Zeichentabelle erste
      llen (Leer)"
[386] 270 LOCATE 10,20:PRINT"<D>irectory *.TBL"
[7E4] 280 LOCATE 12,20:PRINT"<E>nde"
[US6] 290 LOCATE 14,20:PRINT"<?> Auswahl eingeben"
[0H6] 300 LOCATE 14,21,1:A$=INKEY$: IF A$="" THEN 300
[0D2] 310 LOCATE 14,21,0
[1P6] 320 IF A$="L" OR A$="1" THEN GOSUB 370: GOTO 49
      0
[FP6] 330 IF A$="N" OR A$="n" THEN GOSUB 370: GOTO 79
      0
[895] 340 IF A$="E" OR A$="e" THEN CLS: END
[66B] 350 IF A$="D" OR A$="d" THEN CLS: FILES "*.TBL"
      : INPUT "<RETURN>";A$: GOTO 200
[PG2] 360 BEEP: GOTO 300
[DR5] 370 '***** Bildschirm rechts löschen
[1L2] 380 FOR I=5 TO 22
[P35] 390 LOCATE I,20: PRINT SPACE$(60);
[AL1] 400 NEXT I
[1M1] 410 RETURN
[PF5] 420 '***** Raster aufbauen / löschen
[M7A] 430 LOCATE 4,1:PRINT" "
[FK2] 440 FOR I=5 TO 20
[7WB] 450 LOCATE I,1: PRINT"|.....|"
[WL1] 460 NEXT I
[R29] 470 LOCATE 21,1:PRINT" "
[PM1] 480 RETURN
[ZF7] 490 '***** Laden Zeichentabelle (Extension=.TBL)
[173] 500 ON ERROR GOTO 1910
[Y0B] 510 LOCATE 5,20: INPUT "Name der Tabelle zum Lade
      n (max. 8 Zeichen): ";NA$
[Z85] 520 IF NA$="" THEN GOSUB 370: GOTO 250
[VX9] 530 IF LEN(NA$)>8 THEN BEEP: LOCATE 5,20: PRINT S
      PACE$(50): GOTO 510
[WMA] 540 IF INSTR(NA$,".") THEN BEEP: LOCATE 5,20: PRI
      NT SPACE$(50): GOTO 510
[SZ3] 550 NA$=NA$+".TBL": GOSUB 370
[8L2] 560 OPEN "I",#1,NA$
[5T2] 570 FOR A=0 TO 255
[19F] 580 INPUT#1,C(A,1,1),C(A,1,2),C(A,1,3),C(A,1,4),
      C(A,1,5),C(A,1,6),C(A,1,7),C(A,1,8),C(A,1,9),C(A,
      1,10),C(A,1,11),C(A,1,12)
[5AF] 590 INPUT#1,C(A,2,1),C(A,2,2),C(A,2,3),C(A,2,4),
      C(A,2,5),C(A,2,6),C(A,2,7),C(A,2,8),C(A,2,9),C(A,
      2,10),C(A,2,11),C(A,2,12)
[999] 600 LOCATE 5,20:PRINT "Lese Zeichen #";A;" von "
      ;NA$;" ein..."
[1M1] 610 NEXT A
[KM1] 620 CLOSE#1
[S73] 630 ON ERROR GOTO 1970
[BU1] 640 GOSUB 370
[8F6] 650 LOCATE 5,20:PRINT"** EDITIEREN ";NA$;" *"
[UC9] 660 LOCATE 7,20:PRINT"Zeichen mit Pfeiltasten aus
      wählen"
[11D] 670 LOCATE 8,20:PRINT"oder # für Direktwahl über
      ASCII-Code. <E> zum Editieren"
[W9C] 680 LOCATE 9,20:PRINT"<X>=Beenden und Speichern <
      CTRL-Q>=Ende ohne Speichern"
[K68] 690 '***** Aktuelles Zeichen und ASCII-Code ANZEI
      GEN
[DS7] 700 LOCATE 10,20:PRINT"ASCII-CODE: ";CH: POKE 152
      0,CH
[C84] 710 A$=INKEY$: IF A$="" THEN 710
[886] 720 IF A$=LE$ AND CH=0 THEN CH=CH-1: GOTO 690
[CS6] 730 IF A$=RI$ AND CH<255 THEN CH=CH+1: GOTO 690
[HER] 740 IF A$="#" THEN LOCATE 12,20: INPUT "ASCII-COD

```



```
[881] 1930 BEEP
[ GK4] 1940 A$=INKEY$: IF A$="" THEN 1940
[9H7] 1950 LOCATE 21,20: COLOR 7: PRINT"
      "
[M92] 1960 RESUME 510
[RG3] 1970 'FEHLERBEHANDLUNG
[9S9] 1980 LOCATE 21,20: COLOR 15: PRINT"INTERNER FEHLE
      R! <TASTE>"
[Ø91] 1990 BEEP
[UG4] 2000 A$=INKEY$: IF A$="" THEN 2000
[ØG7] 2010 LOCATE 21,20: COLOR 7: PRINT"
      "
[P82] 2020 RESUME 640
```

Listing 2. »prnlq.bas« zum Ausdrucken von Textdateien mit einem beliebigen Zeichensatz.

[MN2]	1: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
[MN2]	2: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
[C65]	3: 126,1,128,160,162,138,138,162,160,128,1,126
[DG4]	4: 126,128,1,69,69,21,21,69,69,1,128,126
[UW5]	5: 126,127,255,223,223,245,245,223,223,255,127,126
[K06]	6: 126,254,255,187,187,235,235,187,187,255,254,126
[P34]	7: 16,16,56,56,28,28,28,56,56,16,16,0
[WV3]	8: 0,48,48,56,56,28,56,56,48,48,0,0
[2G3]	9: 0,0,24,24,60,60,60,24,24,0,0,0
[PX3]	10: 0,16,16,56,56,124,56,56,16,16,0,0
[PT3]	11: 0,0,40,40,120,124,120,40,40,0,0,0
[354]	12: 0,0,24,116,116,124,116,116,24,0,0,0
[CH3]	13: 0,0,24,26,58,62,58,26,24,0,0,0
[PX3]	14: 0,16,16,56,56,124,56,56,16,16,0,0
[CT2]	15: 0,0,0,8,8,28,8,8,0,0,0,0
[222]	16: 0,0,0,0,24,24,24,0,0,0,0,0
[A06]	17: 255,255,255,247,247,227,227,247,247,255,255,255
[0X5]	18: 255,255,255,255,231,231,231,231,255,255,255,255
[S93]	19: 0,8,4,16,16,16,16,16,4,8,0,0
[RX2]	20: 0,8,16,4,4,4,4,16,8,0,0
[MW3]	21: 62,62,34,34,42,34,34,62,62,0,0,0
[EP3]	22: 60,60,36,0,24,0,36,60,60,60,0,0
[LT3]	23: 0,8,4,16,16,16,16,48,16,68,0,0,0
[RV3]	24: 0,8,16,4,4,36,132,196,144,200,0,0
[454]	25: 96,0,144,144,144,156,144,144,144,0,96,0
[XK3]	26: 64,160,0,8,8,28,8,8,0,0,160,64,0
[704]	27: 4,4,4,4,252,128,128,128,128,0,0,0
[7T3]	28: 0,12,12,12,248,64,64,64,64,0,0,0
[LH4]	29: 4,4,4,4,252,128,136,136,136,136,248,0
[194]	30: 0,12,12,12,248,64,64,88,88,88,240,0
[L74]	31: 32,40,32,48,48,112,48,48,32,40,32,0
[UW3]	32: 0,32,48,48,96,96,96,48,48,32,0,0
[PR3]	33: 124,124,56,56,16,16,0,0,0,0,0,0
[RP3]	34: 252,120,120,48,48,0,0,0,0,0,0,0
[KP3]	35: 0,0,0,0,0,16,16,56,56,124,124,0
[HN3]	36: 0,0,0,0,0,48,48,120,120,252,0
[2D3]	37: 0,0,0,64,68,252,68,64,0,0,0,0
[5E3]	38: 0,0,0,8,136,252,136,8,0,0,0,0
[863]	39: 0,0,0,244,0,0,0,244,0,0,0,0
[863]	40: 0,0,0,244,0,0,0,244,0,0,0,0
[ZN4]	41: 0,0,64,32,128,128,128,252,128,128,252,0
[HW3]	42: 0,0,64,128,32,32,32,252,0,0,252,0
[GL4]	43: 0,68,32,144,160,160,160,156,0,64,0
[R04]	44: 8,64,132,36,20,20,20,20,32,136,0,0
[MT2]	45: 4,4,4,4,4,4,4,4,4,4,4,0
[MT2]	46: 4,4,4,4,4,4,4,4,4,4,4,0
[TC3]	47: 0,0,0,72,72,252,72,72,0,0,0,0
[LE3]	48: 4,4,4,4,140,252,140,4,4,4,4,4
[3D3]	49: 0,0,0,64,64,252,64,64,0,0,0,0
[ZC3]	50: 0,0,0,0,128,252,128,0,0,0,0,0
[DV2]	51: 0,0,0,0,4,252,4,0,0,0,0,0
[NX2]	52: 0,0,0,8,8,252,8,8,0,0,0,0
[SU2]	53: 0,0,0,0,0,0,0,36,0,24,0,0
[V14]	54: 16,16,16,16,16,16,16,16,56,16,16,0
[RU2]	55: 0,24,0,36,0,0,0,0,0,0,0,0
[324]	56: 16,16,56,16,16,16,16,16,16,16,16,0
[MP2]	57: 56,0,0,0,0,0,0,0,0,0,0,0
[MV2]	58: 56,8,8,8,8,8,8,8,8,8,0,0
[EB3]	59: 0,24,24,60,0,0,0,60,24,24,0,0
[334]	60: 16,16,56,56,16,16,16,56,56,16,16,0
[9E4]	61: 16,16,48,48,112,112,112,48,48,16,16,0
[RR4]	62: 16,48,48,112,112,240,112,112,48,48,16,0
[DH3]	63: 8,8,12,12,14,14,14,12,12,8,8,0
[524]	64: 16,24,24,28,28,30,28,28,24,24,16,0
[MN2]	65: 0,0,0,0,0,0,0,0,0,0,0,0
[MN2]	66: 0,0,0,0,0,0,0,0,0,0,0,0
[HU2]	67: 0,0,0,0,0,250,0,0,0,0,0,0
[SU2]	68: 0,0,0,0,0,244,0,0,0,0,0,0
[673]	69: 0,0,0,0,192,0,0,192,0,0,0,0
[C73]	70: 0,0,0,0,128,0,0,128,0,0,0,0
[P64]	71: 32,32,32,254,32,32,254,32,32,32,32,0

[F74]	72:	16,16,16,252,16,16,252,16,16,16,0,0
[U34]	73:	32,84,84,254,84,84,254,84,84,8,0,0
[AA3]	74:	96,0,0,252,0,0,252,0,0,24,0,0
[EV3]	75:	4,68,72,8,16,16,32,32,68,196,0,0
[Y74]	76:	4,140,136,16,16,32,32,64,200,136,0,0
[H64]	77:	68,154,162,162,130,210,0,12,0,22,0,0
[YP3]	78:	140,80,32,0,96,0,148,0,24,4,0,0
[N22]	79:	0,0,0,32,0,192,0,0,0,0,0
[L22]	80:	0,0,0,0,64,128,0,0,0,0,0
[163]	81:	56,68,130,0,0,0,0,0,0,0,0
[PY2]	82:	48,204,0,0,0,0,0,0,0,0,0
[Y63]	83:	0,0,0,0,0,0,130,68,56,0,0
[803]	84:	0,0,0,0,0,0,0,204,48,0,0
[2L3]	85:	42,42,8,28,8,62,8,28,8,42,42,0
[3D3]	86:	0,0,36,0,24,126,24,0,36,0,0
[K74]	87:	16,16,16,16,16,124,16,16,16,16,0
[AU2]	88:	0,0,0,0,0,120,0,0,0,0,0
[GP2]	89:	0,0,0,1,0,6,0,0,0,0,0
[8P2]	90:	0,0,0,0,2,4,0,0,0,0,0
[MN2]	91:	0,0,0,0,0,0,0,0,0,0,0
[E84]	92:	16,16,16,16,16,16,16,16,16,16,16
[CP2]	93:	0,0,0,0,4,4,0,0,0,0,0
[CP2]	94:	0,0,0,0,4,4,0,0,0,0,0
[GA3]	95:	0,4,4,8,8,16,16,32,32,64,0,0
[YD3]	96:	0,4,8,8,16,16,32,32,64,64,0,0
[TP4]	97:	124,0,138,130,146,130,162,130,64,124,0,0
[DY3]	98:	120,140,0,16,32,0,64,132,120,0,0
[H73]	99:	0,0,32,2,66,2,254,2,2,2,0,0
[773]	100:	0,0,0,64,0,128,252,0,0,64,0,0
[RU4]	101:	14,130,146,146,146,146,146,146,130,98,0,0
[2A3]	102:	140,16,0,0,0,0,0,32,192,0,0
[RY4]	103:	130,130,130,146,146,146,146,146,146,108,0,0
[203]	104:	0,0,0,0,0,0,0,180,72,0,0
[L73]	105:	16,0,32,0,64,0,252,0,0,0,0,0
[U64]	106:	16,48,16,80,20,148,252,20,20,16,0,0
[YC5]	107:	226,146,146,146,146,146,146,146,128,140,0,0
[AO3]	108:	224,0,0,0,0,0,0,20,8,0,0
[WV4]	109:	124,16,146,146,146,146,146,146,128,12,0,0
[773]	110:	120,132,0,0,0,0,0,20,8,0,0
[TC5]	111:	128,128,128,128,128,134,136,144,160,192,0,0
[S73]	112:	0,0,0,0,4,8,16,32,192,0,0
[0R4]	113:	108,0,146,146,146,146,146,146,108,0,0
[T93]	114:	72,180,0,0,0,0,0,180,72,0,0
[UT4]	115:	96,2,146,146,146,146,146,146,144,124,0,0
[263]	116:	64,160,0,0,0,0,0,4,248,0,0
[P03]	117:	0,0,0,68,68,68,0,0,0,0,0
[XC3]	118:	0,0,0,136,136,136,0,0,0,0,0
[003]	119:	0,0,0,70,68,68,0,0,0,0,0
[VB3]	120:	0,0,0,2,136,140,140,0,0,0,0,0
[622]	121:	0,16,0,40,0,68,0,0,0,0,0
[M63]	122:	0,0,48,0,72,0,132,0,0,0,0,0
[MN2]	123:	0,0,0,0,0,0,0,0,0,0,0
[8Z3]	124:	40,40,40,40,40,40,40,40,40,40,0
[AO3]	125:	0,0,0,0,0,68,0,0,40,0,16,0
[R63]	126:	0,0,0,0,132,0,72,0,48,0,0
[Y74]	127:	0,64,0,128,128,138,128,144,0,96,0,0
[NB3]	128:	0,0,128,0,8,16,0,160,64,0,0
[KU4]	129:	120,4,128,176,136,136,184,136,136,112,0,0
[XY3]	130:	120,128,4,52,68,68,52,4,4,244,0,0
[294]	131:	62,16,80,16,144,144,16,80,16,62,0,0
[CS3]	132:	60,64,0,128,0,0,128,0,64,60,0,0
[515]	133:	254,146,146,146,146,146,146,146,108,0,0
[183]	134:	252,0,0,0,0,0,0,180,72,0,0
[5L4]	135:	124,0,130,130,130,130,130,130,130,0,0
[KA3]	136:	120,132,0,0,0,0,0,132,0,0
[WR4]	137:	254,130,130,130,130,130,130,68,56,0,0
[0B3]	138:	252,0,0,0,0,0,132,0,120,0,0
[9A5]	139:	254,146,146,146,146,146,130,130,130,0,0
[7V2]	140:	252,0,0,0,0,0,0,0,0,0,0
[NC5]	141:	254,144,144,144,144,144,128,128,128,0,0
[7V2]	142:	252,0,0,0,0,0,0,0,0,0,0
[DP4]	143:	124,0,130,130,130,146,146,146,146,30,0,0
[WA3]	144:	120,132,0,0,0,0,0,0,156,0,0
[Y84]	145:	254,16,16,16,16,16,16,16,254,0,0
[K63]	146:	252,0,0,0,0,0,0,0,252,0,0
[TU2]	147:	0,0,0,0,254,0,0,0


```

[7E5] 169: 128,128,128,128,128,254,128,128,128,128,0,0
[NU2] 170: 0,0,0,0,0,252,0,0,0,0,0,0
[P73] 171: 252,0,2,2,2,2,2,2,0,252,0,0
[M83] 172: 248,4,0,0,0,0,0,0,4,248,0,0
[DC3] 173: 224,16,8,4,2,2,4,8,16,224,0,0
[5C3] 174: 224,16,8,4,0,0,4,8,16,224,0,0
[A93] 175: 254,0,4,0,8,8,0,4,0,254,0,0
[U83] 176: 252,4,0,8,0,0,8,0,4,252,0,0
[9S3] 177: 198,0,40,0,16,16,0,40,0,198,0,0
[RS3] 178: 132,72,0,48,0,0,48,0,72,132,0,0
[TV3] 179: 192,32,0,16,14,16,16,0,32,192,0,0
[8G3] 180: 192,0,32,0,28,0,0,32,0,192,0,0
[895] 181: 134,130,138,130,146,130,162,130,194,130,0,0
[G83] 182: 4,8,0,16,0,32,0,64,0,128,0,0
[664] 183: 254,130,130,130,130,130,0,0,0,0,0,0
[V72] 184: 252,0,0,0,0,0,0,0,0,0,0,0
[YT3] 185: 128,64,64,32,32,16,16,8,8,4,4,0
[PZ3] 186: 128,128,64,64,32,32,16,16,8,8,4,0
[664] 187: 0,0,0,0,130,130,130,130,130,254,0,0
[NU2] 188: 0,0,0,0,0,0,0,0,0,252,0,0
[T03] 189: 0,8,0,16,0,32,0,16,0,8,0,0
[463] 190: 0,0,16,0,32,0,32,0,16,0,0,0
[MN2] 191: 0,0,0,0,0,0,0,0,0,0,0,0
[SP2] 192: 1,1,1,1,1,1,1,1,1,1,1,1
[VZ2] 193: 0,0,0,0,192,0,32,0,0,0,0,0
[403] 194: 0,0,0,0,128,64,0,0,0,0,0,0
[NS3] 195: 4,42,42,42,42,42,42,42,10,28,0,0
[9Z2] 196: 12,0,0,0,0,0,0,0,32,28,0,0
[SW3] 197: 254,34,34,34,34,34,34,34,0,28,0,0
[V63] 198: 252,0,0,0,0,0,0,0,36,24,0,0
[GT3] 199: 28,0,34,34,34,34,34,34,34,34,0,0
[ZU2] 200: 24,36,0,0,0,0,0,0,0,0,0,0
[M73] 201: 12,16,2,2,2,2,2,2,2,254,0,0
[BV3] 202: 24,4,32,32,32,32,32,32,252,0,0
[DW3] 203: 28,40,42,42,42,42,42,42,16,0,0
[MU2] 204: 56,4,0,0,0,0,0,0,48,0,0
[RS3] 205: 0,0,0,16,16,126,16,144,128,0,0
[PA3] 206: 0,0,0,0,124,128,0,0,128,0,0
[VT3] 207: 28,0,34,34,34,34,34,34,63,0,0
[H03] 208: 24,37,1,1,1,1,1,1,62,0,0
[S63] 209: 254,0,0,0,0,0,0,16,14,0,0
[GV3] 210: 252,32,32,32,32,32,32,32,0,28,0,0
[6R2] 211: 0,0,0,0,0,62,0,0,0,0,0,0
[0V2] 212: 0,0,0,0,0,188,0,0,0,0,0,0
[HV2] 213: 0,1,0,0,0,0,0,1,190,0,0,0
[WR2] 214: 2,0,1,1,1,1,1,1,62,0,0,0
[6A3] 215: 254,0,8,0,16,8,32,4,64,2,0,0
[8D3] 216: 252,8,0,16,16,32,8,64,4,0,0,0
[XA3] 217: 0,0,0,128,128,254,0,0,0,0,0,0
[NU2] 218: 0,0,0,0,0,252,0,0,0,0,0,0
[RE3] 219: 62,32,32,0,30,0,32,32,0,30,0,0
[2B3] 220: 60,0,0,32,28,32,0,32,28,0,0,0
[EP3] 221: 62,0,32,32,32,32,32,32,0,30,0,0
[L63] 222: 60,32,0,0,0,0,0,32,28,0,0
[RS3] 223: 28,0,34,34,34,34,34,34,28,0,0
[L63] 224: 24,36,0,0,0,0,0,36,24,0,0
[BT3] 225: 63,34,34,34,34,34,34,34,28,0,0
[UZ2] 226: 63,0,0,0,0,0,0,36,24,0,0
[VT3] 227: 28,0,34,34,34,34,34,34,63,0,0
[WZ2] 228: 24,36,0,0,0,0,0,0,63,0,0
[RG3] 229: 62,0,16,0,32,32,32,32,0,16,0,0
[0S3] 230: 60,16,0,32,0,0,0,32,0,0,0
[HW3] 231: 18,42,42,42,42,42,42,42,36,0,0
[SU2] 232: 48,0,0,0,0,0,0,0,12,0,0
[ZV2] 233: 0,0,0,0,252,2,2,2,2,0,0,0
[7B3] 234: 0,0,32,32,252,32,32,0,4,0,0
[AU2] 235: 60,0,2,2,2,2,2,2,0,60,0,0
[XU2] 236: 56,4,0,0,0,0,0,4,56,0,0
[5V2] 237: 56,0,4,0,2,2,0,4,56,0,0
[4W2] 238: 48,8,0,4,0,0,4,0,8,48,0,0
[CV2] 239: 62,0,4,0,8,8,0,4,62,0,0
[2V2] 240: 60,4,0,8,0,0,8,0,4,60,0,0
[H73] 241: 34,0,20,0,8,8,0,20,0,34,0,0
[L53] 242: 0,36,0,24,0,0,24,0,36,0,0,0
[PU2] 243: 60,0,2,2,2,2,2,2,63,0,0
[XU2] 244: 57,5,1,1,1,1,1,1,62,0,0
[CW3] 245: 34,34,38,34,42,42,34,50,34,34,0,0
[TW2] 246: 0,4,0,8,0,0,16,0,32,0,0,0
[UT3] 247: 16,16,108,130,130,130,0,0,0,0,0,0
[V72] 248: 0,0,252,0,0,0,0,0,0,0,0,0
[ZU2] 249: 0,0,0,0,0,238,0,0,0,0,0,0
[MU2] 250: 0,0,0,0,0,204,0,0,0,0,0,0
[MS3] 251: 0,0,0,0,128,128,128,124,0,0,0,0
[473] 252: 0,0,0,0,4,4,4,216,32,32,0,0
[L83] 253: 64,0,128,128,0,64,64,0,128,0,0,0
[YA3] 254: 0,128,0,0,128,0,0,128,0,0,0,0
[G03] 255: 12,16,0,32,0,64,0,32,0,16,12,0
[HF3] 256: 28,4,36,4,68,4,68,4,36,4,28,0
[U93] 257: 0,0,48,2,74,74,74,76,0,0,0,0
[013] 258: 0,0,32,80,0,0,8,4,80,0,0,0
[M93] 259: 188,0,2,2,2,2,2,2,0,188,0,0
[XU2] 260: 56,4,0,0,0,0,0,4,56,0,0
[DW3] 261: 28,40,42,42,42,42,42,42,16,0,0
[VF3] 262: 56,4,0,0,0,0,0,128,128,176,0,0
[3L4] 263: 4,42,170,170,170,170,170,170,28,0,0
[RR3] 264: 140,128,0,0,0,0,0,160,156,0,0
[884] 265: 132,42,42,42,42,42,42,42,158,0,0

```

```

[UT2] 266: 44,0,0,0,0,0,0,0,60,0,0,0
[M84] 267: 132,170,170,42,42,42,42,10,28,0,0
[922] 268: 12,0,0,0,0,0,0,0,32,28,0,0
[B04] 269: 4,42,42,42,170,170,42,42,10,28,0,0
[RF3] 270: 12,0,0,0,128,128,0,0,32,28,0,0
[TZ2] 271: 0,48,8,8,10,10,4,0,0,0,0,0
[LT3] 272: 0,48,64,64,64,64,76,64,64,0,0,0
[3N4] 273: 28,40,170,170,170,170,170,170,42,16,0,0
[SS3] 274: 184,132,0,0,0,0,0,128,176,0,0
[584] 275: 156,40,42,42,42,42,42,42,144,0,0
[MU2] 276: 56,4,0,0,0,0,0,0,48,0,0
[8A4] 277: 156,168,170,42,42,42,42,16,0,0
[MU2] 278: 56,4,0,0,0,0,0,0,48,0,0
[6R2] 279: 0,0,0,0,0,62,0,0,0,0,0,0
[173] 280: 0,0,0,128,0,60,0,128,0,0,0,0
[1D3] 281: 0,0,0,0,128,190,128,0,0,0,0,0
[ET3] 282: 0,0,128,128,0,60,0,128,128,0,0,0
[6R2] 283: 0,0,0,0,0,62,0,0,0,0,0,0
[PT3] 284: 0,0,128,128,128,188,0,0,0,0,0,0
[EC4] 285: 190,16,80,16,144,144,16,80,16,190,0,0
[CS3] 286: 60,64,0,128,0,0,128,0,64,60,0,0
[9W3] 287: 14,24,8,40,136,136,40,8,24,30,0,0
[3D3] 288: 28,0,32,0,64,64,0,32,0,28,0,0
[P44] 289: 62,10,10,10,10,10,2,130,130,0,0,0
[W34] 290: 124,64,64,64,64,64,64,64,64,0,0
[4T3] 291: 4,42,42,42,42,42,62,42,42,42,0,0
[EU2] 292: 12,0,0,0,0,60,0,0,0,0,0,0
[BP4] 293: 30,16,48,16,80,16,254,146,146,146,130,0
[2G3] 294: 28,32,0,64,0,128,252,0,0,0,0,0
[MD4] 295: 28,0,162,162,162,162,162,162,0,28,0,0
[YR3] 296: 152,164,0,0,0,0,0,0,164,152,0,0
[DX3] 297: 156,0,34,34,34,34,34,34,0,156,0,0
[L63] 298: 24,36,0,0,0,0,0,36,24,0,0
[Z94] 299: 156,128,162,34,34,34,34,34,28,0,0
[L63] 300: 24,36,0,0,0,0,0,36,24,0,0
[4A4] 301: 60,0,130,130,130,130,130,130,60,0,0,0
[YR3] 302: 184,132,0,0,0,0,0,132,184,0,0
[AU2] 303: 60,0,2,2,2,2,2,2,60,0,0,0
[DK3] 304: 184,132,128,0,0,0,0,0,4,56,0,0
[393] 305: 188,0,2,2,2,2,2,2,191,0,0
[XU2] 306: 57,5,1,1,1,1,1,1,62,0,0,0
[P04] 307: 188,0,66,66,66,66,66,66,0,188,0,0
[583] 308: 56,68,0,0,0,0,0,68,56,0,0,0
[M93] 309: 188,0,2,2,2,2,2,2,0,188,0,0
[463] 310: 120,4,0,0,0,0,0,4,120,0,0
[FZ2] 311: 56,0,0,0,0,198,0,0,0,0,0,0
[U24] 312: 48,72,72,72,72,204,72,72,72,72,0,0
[VS3] 313: 2,2,62,66,2,130,130,130,2,66,0,0,0
[7R3] 314: 32,36,124,32,160,0,0,0,128,0,0,0
[B94] 315: 208,48,16,16,30,16,16,16,48,208,0,0
[4K3] 316: 200,8,40,8,28,8,8,40,8,200,0,0
[515] 317: 254,128,128,128,128,128,140,34,66,0,0
[774] 318: 252,32,32,32,32,40,40,44,136,72,0,0
[US3] 319: 0,0,1,17,16,126,16,144,128,0,0,0
[2B3] 320: 0,2,0,0,2,124,128,0,0,128,0,0,0
[D94] 321: 4,42,42,42,42,42,42,170,138,156,0,0
[9Z2] 322: 12,0,0,0,0,0,0,0,32,28,0,0
[6R2] 323: 0,0,0,0,0,62,0,0,0,0,0,0
[XF3] 324: 0,0,0,0,0,60,128,128,128,0,0,0
[W94] 325: 28,0,34,34,34,34,34,162,128,156,0,0
[L63] 326: 24,36,0,0,0,0,0,36,24,0,0
[FF3] 327: 60,0,2,2,2,2,2,130,128,188,0,0
[XU2] 328: 56,4,0,0,0,0,0,4,56,0,0
[Z25] 329: 190,128,160,160,160,160,160,128,30,0,0
[L63] 330: 60,32,0,0,0,0,0,32,28,0,0,0
[YB5] 331: 190,160,128,144,128,136,128,132,128,190,0,0
[NB3] 332: 124,0,32,0,16,0,8,0,4,124,0,0
[E73] 333: 18,42,2,2,2,34,26,2,2,2,0,0,0
[4B3] 334: 48,0,72,72,72,8,56,8,8,8,0,0
[783] 335: 18,42,2,2,2,2,2,2,42,18,0,0
[VT3] 336: 48,0,72,72,72,72,72,72,48,0,0
[V73] 337: 4,0,10,2,18,162,2,2,0,4,0,0,0
[FZ2] 338: 0,12,0,16,0,96,0,0,4,0,0,0,0
[0P2] 339: 12,0,0,0,0,0,0,0,0,0,0,0
[5X3] 340: 28,16,16,16,16,16,16,16,16,16,0,0
[2R2] 341: 0,0,0,0,0,0,0,0,12,0,0,0
[2X3] 342: 16,16,16,16,16,16,16,16,28,0,0,0
[NT3] 343: 0,68,100,8,8,16,16,36,32,64,0,8
[724] 344: 0,36,232,40,16,16,32,36,92,92,28,4
[KK3] 345: 0,4,100,8,8,16,20,36,44,76,0,0
[8S3] 346: 0,4,200,8,16,16,32,40,64,92,0,0
[KU2] 347: 0,0,0,0,0,222,0,0,0,0,0,0,0
[0V2] 348: 0,0,0,0,0,188,0,0,0,0,0,0,0
[283] 349: 0,16,0,0,0,84,0,0,0,0,68,0,0
[PG3] 350: 0,0,48,0,72,0,180,0,72,0,132,0
[W83] 351: 0,0,68,0,40,0,84,0,40,0,16,0
[9G3] 352: 0,132,0,72,0,180,0,72,0,48,0,0
[HC3] 353: 255,0,0,0,255,0,0,0,255,0,0,0

```

Listing 3. Die Zeichensatzdatei »standard.tbl«


```

[HC3] 354: 0,0,255,0,0,0,255,0,0,0,255,0
[E84] 355: 0,255,0,255,0,255,0,255,0,255,0,255
[E84] 356: 255,0,255,0,255,0,255,0,255,0,255,0
[8U4] 357: 146,36,73,146,36,73,146,36,73,146,36,73
[8U4] 358: 73,146,36,73,146,36,73,146,36,73,146,36
[VU2] 359: 0,0,0,0,0,255,0,0,0,0,0,0
[VU2] 360: 0,0,0,0,0,255,0,0,0,0,0,0
[BY2] 361: 8,8,8,8,8,255,0,0,0,0,0,0
[VU2] 362: 0,0,0,0,0,255,0,0,0,0,0,0
[RF3] 363: 24,24,24,24,24,255,0,0,0,0,0,0
[VU2] 364: 0,0,0,0,0,255,0,0,0,0,0,0
[6S3] 365: 24,24,24,24,24,255,0,255,0,0,0,0
[K63] 366: 0,0,0,0,0,255,0,255,0,0,0,0
[SY2] 367: 8,8,8,8,8,15,8,15,0,0,0,0
[GU2] 368: 0,0,0,0,0,15,0,15,0,0,0,0
[9B3] 369: 24,24,24,24,24,31,0,0,0,0,0,0
[TP2] 370: 0,0,0,0,0,31,0,0,0,0,0,0
[6S3] 371: 24,24,24,24,24,255,0,0,0,0,0,0
[S63] 372: 0,0,0,0,0,239,0,255,0,0,0,0
[K63] 373: 0,0,0,0,0,255,0,255,0,0,0,0
[K63] 374: 0,0,0,0,0,255,0,255,0,0,0,0
[CP3] 375: 24,24,24,24,24,31,16,31,0,0,0,0
[AU2] 376: 0,0,0,0,0,15,0,31,0,0,0,0
[MS3] 377: 24,24,24,24,24,248,8,248,0,0,0,0
[P53] 378: 0,0,0,0,0,224,0,240,0,0,0,0
[GB3] 379: 8,8,8,8,8,248,8,248,0,0,0,0
[G53] 380: 0,0,0,0,0,240,0,240,0,0,0,0
[YF3] 381: 24,24,24,24,24,248,0,0,0,0,0,0
[GU2] 382: 0,0,0,0,0,240,0,0,0,0,0,0
[SU2] 383: 8,8,8,8,8,15,0,0,0,0,0,0
[NP2] 384: 0,0,0,0,0,15,0,0,0,0,0,0
[GZ2] 385: 0,0,0,0,0,248,8,8,8,8,8,8
[GU2] 386: 0,0,0,0,0,240,0,0,0,0,0,0
[Q33] 387: 8,8,8,8,8,248,8,8,8,8,8
[GU2] 388: 0,0,0,0,0,240,0,0,0,0,0,0
[5X2] 389: 8,8,8,8,8,15,8,8,8,8,8
[NP2] 390: 0,0,0,0,0,15,0,0,0,0,0,0
[BZ2] 391: 0,0,0,0,0,255,8,8,8,8,8,8
[VU2] 392: 0,0,0,0,0,255,0,0,0,0,0,0
[MY2] 393: 8,8,8,8,8,8,8,8,8,8,8
[MN2] 394: 0,0,0,0,0,0,0,0,0,0,0
[V23] 395: 8,8,8,8,8,255,8,8,8,8,8
[VU2] 396: 0,0,0,0,0,255,0,0,0,0,0,0
[XP3] 397: 0,0,0,0,0,255,24,24,24,24,24,24
[VU2] 398: 0,0,0,0,0,255,0,0,0,0,0,0
[K93] 399: 0,0,0,0,0,255,0,255,8,8,8,8
[K63] 400: 0,0,0,0,0,255,0,255,0,0,0,0
[MS3] 401: 0,0,0,0,0,248,8,248,24,24,24,24
[U53] 402: 0,0,0,0,0,240,0,224,0,0,0,0
[XG3] 403: 0,0,0,0,0,31,16,31,24,24,24,24
[MU2] 404: 0,0,0,0,0,31,0,15,0,0,0,0
[S94] 405: 24,24,24,24,24,248,8,248,24,24,24,24
[Q63] 406: 0,0,0,0,0,224,0,224,0,0,0,0
[Q74] 407: 24,24,24,24,24,31,16,31,24,24,24,24
[GU2] 408: 0,0,0,0,0,15,0,15,0,0,0,0
[6R3] 409: 0,0,0,0,0,255,0,255,24,24,24,24
[W63] 410: 0,0,0,0,0,255,0,239,0,0,0,0
[G74] 411: 24,24,24,24,24,24,24,24,24,24,24,24
[MN2] 412: 0,0,0,0,0,0,0,0,0,0,0,0
[A94] 413: 24,24,24,24,24,255,0,255,24,24,24,24
[Z73] 414: 0,0,0,0,0,239,0,239,0,0,0,0
[B94] 415: 24,24,24,24,24,24,24,248,24,24,24,24
[PU2] 416: 0,0,0,0,0,0,0,224,0,0,0,0
[GE3] 417: 8,8,8,8,8,248,8,248,8,8,8,8
[G53] 418: 0,0,0,0,0,240,0,240,0,0,0,0
[B74] 419: 24,24,24,24,24,31,24,24,24,24,24,24
[NP2] 420: 0,0,0,0,0,15,0,0,0,0,0,0
[S13] 421: 8,8,8,8,8,15,8,15,8,8,8,8
[GU2] 422: 0,0,0,0,0,15,0,15,0,0,0,0
[Q63] 423: 0,0,0,0,0,248,8,248,8,8,8,8
[G53] 424: 0,0,0,0,0,240,0,240,0,0,0,0
[ZR3] 425: 0,0,0,0,0,248,24,24,24,24,24,24
[GU2] 426: 0,0,0,0,0,240,0,0,0,0,0,0
[ZG3] 427: 0,0,0,0,0,31,24,24,24,24,24,24
[TP2] 428: 0,0,0,0,0,31,0,0,0,0,0,0
[8Y2] 429: 0,0,0,0,0,15,8,15,8,8,8,8
[GU2] 430: 0,0,0,0,0,15,0,15,0,0,0,0
[ZE3] 431: 8,8,8,8,8,255,8,255,8,8,8,8
[K63] 432: 0,0,0,0,0,255,0,255,0,0,0,0
[X84] 433: 24,24,24,24,24,255,24,24,24,24,24,24
[VU2] 434: 0,0,0,0,0,255,0,0,0,0,0,0
[GY2] 435: 8,8,8,8,8,248,0,0,0,0,0,0
[GU2] 436: 0,0,0,0,0,240,0,0,0,0,0,0
[NT2] 437: 0,0,0,0,0,15,8,8,8,8,8,8
[NP2] 438: 0,0,0,0,0,15,0,0,0,0,0,0
[SZ5] 439: 255,255,255,255,255,255,255,255,255,255,255
[SZ5] 440: 255,255,255,255,255,255,255,255,255,255,255
[G74] 441: 15,15,15,15,15,15,15,15,15,15,15
[G74] 442: 15,15,15,15,15,15,15,15,15,15,15
[PA4] 443: 255,255,255,255,255,255,0,0,0,0,0,0
[PA4] 444: 255,255,255,255,255,255,0,0,0,0,0,0
[DB4] 445: 0,0,0,0,0,255,255,255,255,255,255,255
[DB4] 446: 0,0,0,0,0,255,255,255,255,255,255,255
[4R5] 447: 240,240,240,240,240,240,240,240,240,240,240

```

```

[WU5] 448: 224,224,224,224,224,224,224,224,224,224,224
[NV2] 449: 12,2,2,2,2,10,0,4,8,2,0,0
[HH3] 450: 12,16,16,16,16,16,12,8,4,16,0,0
[4R4] 451: 127,0,128,146,146,146,146,146,0,108,0,0
[PH3] 452: 127,128,0,0,0,0,0,180,72,0,0
[883] 453: 0,0,0,0,62,32,32,32,48,0,0,0
[TU2] 454: 0,0,0,0,60,0,0,0,32,0,0,0
[P73] 455: 16,16,16,0,14,0,8,0,14,0,0,0
[4D3] 456: 0,0,0,16,28,16,0,16,28,16,0,0
[385] 457: 130,198,130,170,130,146,130,130,130,130,0,0
[CB3] 458: 132,0,72,0,48,0,0,0,132,0,0,0
[QX3] 459: 28,0,34,34,34,34,34,32,60,32,32,0
[563] 460: 24,36,0,0,0,0,0,36,24,0,0,0
[7W2] 461: 28,3,2,2,2,2,2,0,28,0,0,0
[ZU2] 462: 25,6,0,0,0,0,0,4,24,0,0,0
[S53] 463: 0,0,16,16,16,14,0,0,16,0,0,0
[873] 464: 0,16,0,0,16,12,16,16,0,0,0,0
[VS4] 465: 0,130,146,170,130,198,130,170,146,130,0,0
[MD3] 466: 0,0,48,0,72,132,72,0,48,0,0,0
[ZH3] 467: 0,0,16,56,16,84,16,56,16,0,0,0
[L73] 468: 0,0,48,0,72,0,72,0,48,0,0,0
[DW3] 469: 18,10,38,32,32,32,32,38,10,18,0,0
[DA3] 470: 16,32,12,0,0,0,0,12,32,16,0,0
[B23] 471: 4,8,66,2,34,2,18,2,8,4,0,0
[A14] 472: 8,4,16,208,144,176,144,144,4,8,0,0
[GT3] 473: 8,16,16,16,16,8,16,16,16,16,8,0
[743] 474: 16,8,8,8,8,16,8,8,8,16,0,0
[MT3] 475: 8,16,18,16,20,8,16,48,16,16,8,0
[ME3] 476: 16,8,8,12,8,16,4,0,8,72,8,16,0
[1W3] 477: 0,0,0,56,16,84,16,146,146,130,0,0
[G63] 478: 0,0,0,48,72,0,132,0,0,0,0,0
[8R4] 479: 126,0,128,128,128,128,128,0,126,0,0,0
[4S3] 480: 124,128,0,0,0,0,0,128,124,0,0,0
[FY3] 481: 18,18,18,18,18,18,18,18,18,18,0,0
[4X2] 482: 8,8,8,8,8,8,8,8,8,8,0,0
[LY3] 483: 18,18,18,18,18,58,18,18,18,18,0,0
[AU2] 484: 0,0,0,0,0,120,0,0,0,0,0,0
[DW2] 485: 2,2,2,2,2,42,2,18,2,2,0,0
[4W2] 486: 0,0,0,0,72,0,48,0,0,0,0,0
[WW2] 487: 2,2,2,2,18,2,42,2,2,2,0,0
[ZU2] 488: 0,0,0,0,0,48,0,72,0,0,0,0
[EC3] 489: 0,0,0,0,0,126,0,128,128,0,0,0
[PA3] 490: 0,0,0,0,0,124,128,0,0,128,0,0
[9V2] 491: 4,0,2,2,0,252,0,0,0,0,0,0
[SV2] 492: 0,4,0,0,4,248,0,0,0,0,0,0
[7V2] 493: 0,0,0,0,36,36,0,0,0,0,0,0
[8X3] 494: 16,16,16,16,84,84,16,16,16,16,0,0
[K83] 495: 4,4,4,32,32,4,4,4,32,32,0,0
[5D3] 496: 32,32,32,8,8,32,32,32,8,0,0,0
[BR3] 497: 0,0,0,64,128,128,128,64,0,0,0,0
[H33] 498: 0,0,0,128,64,64,64,128,0,0,0,0
[DV2] 499: 0,0,0,0,0,16,16,0,0,0,0,0
[DV2] 500: 0,0,0,0,0,16,16,0,0,0,0,0
[MN2] 501: 0,0,0,0,0,0,0,0,0,0,0,0
[4R2] 502: 0,0,0,0,0,16,0,0,0,0,0,0
[TB3] 503: 0,0,0,0,8,0,4,0,2,254,128,128
[3D3] 504: 16,16,16,16,0,0,8,0,4,0,252,0,0
[EU3] 505: 224,128,128,128,96,0,0,0,0,0,0,0
[Q63] 506: 224,0,0,0,224,0,0,0,0,0,0,0
[S63] 507: 0,160,128,64,0,0,0,0,0,0,0,0
[ZA3] 508: 160,32,96,160,0,0,0,0,0,0,0,0
[863] 509: 0,0,0,0,24,24,24,24,0,0,0,0
[K63] 510: 0,0,0,0,16,16,16,16,0,0,0,0
[MN2] 511: 0,0,0,0,0,0,0,0,0,0,0,0
[MN2] 512: 0,0,0,0,0,0,0,0,0,0,0,0

```

Listing 3. Die Zeichensatzdatei »standard.tbl«

```

[MN2] 1: 0,0,0,0,0,0,0,0,0,0,0,0
[MN2] 2: 0,0,0,0,0,0,0,0,0,0,0,0
[C65] 3: 126,1,128,160,162,138,138,162,160,128,1,126
[DG4] 4: 126,128,1,69,69,21,21,69,69,1,128,126
[UW5] 5: 126,127,255,223,223,245,245,223,223,255,127
[K06] 6: 126,254,255,187,187,235,235,187,187,255,254
[P34] 7: 16,16,56,56,28,28,28,56,56,16,16,0
[WV3] 8: 0,48,48,56,56,28,56,56,48,48,0,0
[ZG3] 9: 0,0,24,24,60,60,60,24,24,0,0,0
[PX3] 10: 0,16,16,56,56,124,56,56,16,16,0,0
[PT3] 11: 0,0,40,40,120,124,120,40,40,0,0,0
[354] 12: 0,0,24,116,116,124,116,116,24,0,0,0
[CH3] 13: 0,0,24,26,58,62,58,26,24,0,0,0
[PX3] 14: 0,16,16,56,56,124,56,56,16,16,0,0
[CT2] 15: 0,0,0,8,8,28,8,8,0,0,0,0
[Z22] 16: 0,0,0,0,24,24,24,0,0,0,0,0
[A06] 17: 255,255,255,255,231,231,231,231,255,255,255
[QX5] 18: 255,255,255,255,231,231,231,231,255,255,255

```



```

[S93] 19: 0,8,4,16,16,16,16,16,4,8,0,0
[RX2] 20: 0,8,16,4,4,4,4,16,8,0,0
[MW3] 21: 62,62,34,34,42,34,34,62,62,0,0
[EP3] 22: 60,60,36,0,24,0,36,60,60,0,0
[LT3] 23: 0,8,4,16,16,16,48,16,68,104,0,0
[RV3] 24: 0,8,16,4,4,36,132,196,144,200,0,0
[4S4] 25: 96,0,144,144,144,156,144,144,144,96,0
[XK3] 26: 64,160,0,8,8,28,8,8,0,160,64,0
[704] 27: 4,4,4,4,252,128,128,128,128,0,0,0
[7T3] 28: 0,12,12,12,248,64,64,64,64,0,0,0
[LH4] 29: 4,4,4,4,252,128,136,136,136,136,248,0
[194] 30: 0,12,12,12,248,64,64,88,88,240,0
[C74] 31: 40,40,32,48,48,112,48,48,32,40,40,0
[UW3] 32: 0,32,48,48,96,96,96,48,48,32,0,0
[PR3] 33: 124,124,56,56,16,16,0,0,0,0,0,0
[RP3] 34: 252,120,120,48,48,0,0,0,0,0,0
[KP3] 35: 0,0,0,0,0,16,16,56,56,124,124,0
[HN3] 36: 0,0,0,0,0,0,48,48,120,120,252,0
[ZD3] 37: 0,0,0,64,68,252,68,64,0,0,0,0
[5E3] 38: 0,0,0,8,136,252,136,8,0,0,0,0
[863] 39: 0,0,0,244,0,0,0,244,0,0,0,0
[863] 40: 0,0,0,244,0,0,0,244,0,0,0,0
[ZN4] 41: 0,0,64,32,128,128,128,252,128,128,252,0
[HW3] 42: 0,0,64,128,32,32,32,252,0,0,252,0
[GL4] 43: 0,68,32,144,160,160,160,160,156,0,64,0
[R04] 44: 8,64,132,36,20,20,20,20,32,136,0,0
[MT2] 45: 4,4,4,4,4,4,4,4,4,4,4,0
[MT2] 46: 4,4,4,4,4,4,4,4,4,4,4,0
[TC3] 47: 0,0,0,72,72,252,72,72,0,0,0,0
[LE3] 48: 4,4,4,4,140,252,140,4,4,4,4,4
[3D3] 49: 0,0,0,64,64,252,64,64,0,0,0,0
[ZC3] 50: 0,0,0,0,128,252,128,0,0,0,0,0
[DV2] 51: 0,0,0,0,4,252,4,0,0,0,0,0
[NX2] 52: 0,0,0,8,8,252,8,8,0,0,0,0
[SU2] 53: 0,0,0,0,0,0,0,36,0,24,0,0
[V14] 54: 16,16,16,16,16,16,16,56,16,16,0
[RU2] 55: 0,24,0,36,0,0,0,0,0,0,0,0
[324] 56: 16,16,56,16,16,16,16,16,16,16,0
[MP2] 57: 56,0,0,0,0,0,0,0,0,0,0,0
[MV2] 58: 56,8,8,8,8,8,8,8,8,8,0,0
[EB3] 59: 0,24,24,60,0,0,0,60,24,24,0,0
[334] 60: 16,16,56,56,16,16,16,56,16,16,0
[9E4] 61: 16,16,48,48,112,112,112,48,48,16,16,0
[RR4] 62: 16,48,48,112,112,240,112,112,48,48,16,0
[DH3] 63: 8,8,12,12,14,14,14,12,12,8,8,0
[524] 64: 16,24,24,28,28,30,28,28,24,24,16,0
[MN2] 65: 0,0,0,0,0,0,0,0,0,0,0,0
[MN2] 66: 0,0,0,0,0,0,0,0,0,0,0,0
[SU2] 67: 0,0,0,0,0,244,0,0,0,0,0,0
[SU2] 68: 0,0,0,0,0,244,0,0,0,0,0,0
[673] 69: 0,0,0,0,192,0,192,0,0,0,0,0
[C73] 70: 0,0,0,0,128,0,0,128,0,0,0,0
[984] 71: 32,32,32,252,32,32,32,252,32,32,0,0
[T94] 72: 16,16,16,252,16,16,16,252,16,16,0,0
[9A4] 73: 32,80,80,252,80,80,80,252,80,80,72,0
[ON3] 74: 104,8,8,252,8,8,8,252,8,8,16,0
[T34] 75: 4,68,72,8,16,16,32,32,68,196,128,0
[ZD4] 76: 4,140,136,16,16,32,32,64,72,136,128,0
[F84] 77: 68,152,160,160,128,148,64,8,0,20,0,0
[N84] 78: 136,84,100,68,100,64,152,0,24,0,36,0
[NZ2] 79: 0,0,0,0,32,0,192,0,0,0,0,0
[LZ2] 80: 0,0,0,0,0,64,128,0,0,0,0,0
[373] 81: 16,40,68,130,0,0,0,0,0,0,0,0
[Y53] 82: 48,72,132,0,0,0,0,0,0,0,0,0
[B63] 83: 0,0,0,0,0,0,0,132,72,48,0,0
[B63] 84: 0,0,0,0,0,0,0,132,72,48,0,0
[AF3] 85: 8,42,42,8,28,8,28,8,42,42,8,0
[GZ2] 86: 0,0,0,36,0,24,0,36,0,0,0,0
[K74] 87: 16,16,16,16,16,16,124,16,16,16,16,0
[NU2] 88: 0,0,0,0,0,252,0,0,0,0,0,0
[GP2] 89: 0,0,0,1,0,6,0,0,0,0,0,0
[8P2] 90: 0,0,0,0,2,4,0,0,0,0,0,0
[L14] 91: 16,16,16,16,16,16,16,16,16,16,0
[MN2] 92: 0,0,0,0,0,0,0,0,0,0,0,0
[MP2] 93: 0,0,0,0,4,4,4,0,0,0,0,0
[MP2] 94: 0,0,0,0,4,4,4,0,0,0,0,0
[YE3] 95: 0,4,4,8,16,16,32,32,64,64,0,0
[6E3] 96: 4,4,8,8,16,16,32,32,64,64,0,0
[KU4] 97: 120,4,128,144,144,128,128,160,160,4,120,0
[314] 98: 120,144,20,4,4,36,36,4,4,192,120,0
[B63] 99: 0,0,32,0,64,0,252,0,0,0,0,0
[F84] 100: 0,0,4,68,4,132,252,4,4,4,4,0
[HR4] 101: 68,0,136,128,144,144,128,160,0,64,0
[YC3] 102: 4,140,4,20,4,4,4,36,4,196,4,0
[WU4] 103: 4,128,144,144,144,144,144,144,4,104,0
[LC3] 104: 128,4,4,4,4,4,4,4,176,72,0
[L73] 105: 16,0,32,0,64,0,252,0,0,0,0,0
[U84] 106: 16,48,16,80,20,148,252,20,0,16,16,0
[HA5] 107: 100,144,144,144,144,144,144,144,132,8,0
[3B3] 108: 224,4,4,4,4,4,4,4,16,136,0
[8Y4] 109: 120,20,144,144,144,144,144,144,132,8,0
[VK3] 110: 120,128,4,4,4,4,4,4,16,136,0
[EK5] 111: 192,128,128,128,128,132,136,144,160,128,192,0
[AK3] 112: 128,0,0,0,0,4,8,16,32,64,128,0
[0V4] 113: 104,4,144,144,144,144,144,144,4,104,0
[FD3] 114: 72,176,4,4,4,4,4,4,176,72,0

[ZW4] 115: 96,4,144,144,144,144,144,144,144,20,120,0
[3G3] 116: 72,160,4,4,4,4,4,4,128,120,0
[P03] 117: 0,0,0,0,68,68,68,0,0,0,0,0
[XC3] 118: 0,0,0,0,136,136,136,0,0,0,0,0
[003] 119: 0,0,0,0,0,68,68,0,0,0,0,0
[VB3] 120: 0,0,0,2,136,140,140,0,0,0,0,0
[622] 121: 0,16,0,40,0,68,0,0,0,0,0,0
[M63] 122: 0,0,48,0,72,0,132,0,0,0,0,0
[MN2] 123: 0,0,0,0,0,0,0,0,0,0,0,0
[8Z3] 124: 40,40,40,40,40,40,40,40,40,40,0,0
[A03] 125: 0,0,0,0,0,68,0,40,0,16,0,0
[R63] 126: 0,0,0,0,0,132,0,72,0,48,0,0
[S74] 127: 0,64,0,128,128,136,128,144,0,96,0,0
[CG3] 128: 64,0,128,0,0,4,16,0,160,64,0,0
[TV4] 129: 120,4,128,176,136,136,136,184,136,8,120,0
[AB4] 130: 120,128,4,52,68,68,68,52,4,132,116,0
[YE3] 131: 0,4,8,48,80,144,80,48,8,4,0,0
[5V3] 132: 4,12,20,32,64,128,64,32,20,12,4,0
[6D5] 133: 128,128,252,128,128,128,128,128,52,72,0
[P34] 134: 4,4,252,36,36,36,36,36,128,88,0
[LP4] 135: 48,72,4,128,128,128,128,128,4,64,0
[RE3] 136: 48,72,128,4,4,4,4,4,128,8,0
[635] 137: 128,128,252,128,128,128,128,4,64,56,0
[UE3] 138: 4,4,252,4,4,4,4,4,128,8,112,0
[HL5] 139: 128,128,252,128,128,128,128,176,128,128,132,0
[EU3] 140: 4,4,252,36,36,36,36,4,4,132,0
[DK5] 141: 128,128,252,144,144,144,144,144,128,128,128,0
[VG3] 142: 132,4,252,4,4,0,0,48,0,0,128,0
[AV4] 143: 48,72,4,128,128,128,144,144,144,16,92,0
[8K3] 144: 48,72,128,4,4,4,4,4,132,28,0
[BU4] 145: 128,252,144,16,16,16,16,144,252,128,0
[G73] 146: 4,252,4,0,0,0,0,0,4,252,4,0
[FG4] 147: 0,0,128,128,128,252,128,128,0,0,0,0
[XW2] 148: 0,0,4,4,4,252,4,4,4,0,0,0
[RZ3] 149: 8,4,0,0,0,0,128,132,248,128,128,0
[4Y2] 150: 8,0,4,4,4,4,4,0,248,0,0,0
[VK4] 151: 128,128,252,16,16,0,40,0,196,128,128,0
[RD3] 152: 4,4,252,0,0,48,0,72,4,132,4,0
[FZ3] 153: 128,128,252,128,128,0,0,0,4,4,0,0
[133] 154: 4,4,252,4,4,4,4,4,4,12,0
[TE4] 155: 128,252,192,0,32,0,32,0,192,252,128,0
[CH3] 156: 4,252,4,64,0,32,0,64,4,252,4,0
[XE4] 157: 128,252,192,0,32,0,16,0,136,252,128,0
[UG3] 158: 4,252,4,64,0,32,0,16,4,252,4,0
[9K4] 159: 48,72,4,128,128,128,128,128,4,72,48,0
[NT3] 160: 48,72,128,4,4,4,4,4,128,72,48,0
[165] 161: 128,128,252,144,144,144,144,144,0,96,0
[Z93] 162: 4,4,252,4,4,0,0,0,160,64,0
[1H4] 163: 48,72,4,128,128,144,128,140,0,76,48,0
[4U3] 164: 48,72,128,4,4,4,20,0,136,64,52,0
[465] 165: 128,128,252,144,144,144,152,144,4,96,0
[MH3] 166: 4,4,252,4,4,0,16,0,12,164,68,0
[LY4] 167: 100,0,144,144,144,144,144,144,132,72,0
[7D3] 168: 72,164,4,4,4,4,4,4,16,136,0
[TL5] 169: 192,128,128,128,128,252,128,128,128,192,0
[FE3] 170: 128,0,0,4,4,252,4,4,0,0,128,0
[W74] 171: 128,248,132,0,0,0,0,0,132,248,128,0
[S83] 172: 0,248,0,4,4,4,4,4,0,248,0,0
[484] 173: 128,240,136,0,4,0,4,0,136,240,128,0
[S63] 174: 0,240,0,8,0,4,0,8,0,240,0,0
[2A4] 175: 128,252,128,8,0,16,0,8,128,252,128,0
[KP3] 176: 4,252,12,0,16,0,16,0,12,252,4,0
[7M4] 177: 128,196,128,4,0,0,16,0,40,128,196,128,0
[NS3] 178: 4,132,76,0,48,0,48,0,76,132,4,0
[XK4] 179: 128,192,160,0,16,12,16,0,160,192,128,0
[ZH3] 180: 0,192,0,36,4,28,4,36,0,192,0,0
[BH5] 181: 192,132,144,152,144,144,176,144,208,128,132,0
[YS3] 182: 132,4,12,4,20,36,4,68,4,132,12,0
[FA4] 183: 252,128,128,128,128,0,0,0,0,0,0,0
[ZW2] 184: 252,4,4,4,4,4,0,0,0,0,0,0
[YT3] 185: 128,64,64,32,32,16,16,8,8,4,4,0
[PZ3] 186: 128,128,64,64,32,32,16,16,8,8,4,0
[V54] 187: 0,0,0,0,0,128,128,128,128,252,0,0
[7X2] 188: 0,0,0,0,0,4,4,4,4,252,0,0
[T03] 189: 0,8,0,16,0,32,0,32,0,16,0,0,0
[463] 190: 0,0,16,0,32,0,32,0,16,0,0,0
[MN2] 191: 0,0,0,0,0,0,0,0,0,0,0,0
[SP2] 192: 1,1,1,1,1,1,1,1,1,1,1,1
[V22] 193: 0,0,0,0,192,0,32,0,0,0,0,0
[403] 194: 0,0,0,0,128,64,0,0,0,0,0,0
[N53] 195: 12,32,0,0,0,0,0,0,0,36,24,0
[B44] 196: 40,20,84,84,84,84,84,84,16,60,0
[S53] 197: 128,128,252,0,0,0,8,0,0,36,24,0
[YX3] 198: 4,4,248,68,68,68,68,68,0,56,0

```

Listing 4. Die Zeichensatzdatei »roman.tbl«

[TZ2] 199: 24,36,0,0,0,0,0,0,0,32,0
 [8Y3] 200: 56,0,68,68,68,68,68,68,68,4,0
 [2S3] 201: 24,36,0,0,0,0,0,128,128,252,0,0
 [G14] 202: 56,0,68,68,68,68,68,68,68,248,4,0
 [863] 203: 24,36,0,0,0,0,0,0,0,36,16,0
 [254] 204: 56,16,84,84,84,84,84,84,16,40,0
 [MR3] 205: 0,0,0,0,0,124,0,128,128,64,64,0
 [GU3] 206: 0,0,4,36,36,124,164,36,4,128,0,0
 [Z53] 207: 25,36,0,0,0,0,0,0,0,33,62,0
 [G24] 208: 56,1,69,69,69,69,69,69,4,126,0
 [394] 209: 128,128,252,32,32,32,32,32,0,28,0,0
 [L83] 210: 4,4,252,4,0,0,0,0,36,28,4,0
 [Y73] 211: 0,0,0,0,128,188,0,0,0,0,0,0
 [U83] 212: 0,0,0,68,68,124,4,4,0,0,0,0
 [YW2] 213: 0,2,3,0,0,0,0,65,94,0,0,0
 [M63] 214: 0,2,0,1,1,33,33,32,62,0,0,0
 [6C4] 215: 128,128,252,16,16,16,0,40,96,68,64,0
 [XC3] 216: 4,4,252,0,0,0,32,48,0,12,68,4,0
 [RN3] 217: 0,0,0,128,128,128,252,0,0,0,0,0
 [FX2] 218: 0,0,0,4,4,4,252,4,4,4,0,0
 [P83] 219: 0,60,32,0,0,0,60,0,0,0,32,28,0
 [TY3] 220: 68,124,4,64,64,60,68,64,64,4,60,0
 [973] 221: 0,28,32,0,0,0,0,0,32,28,0,0
 [LV3] 222: 68,124,4,64,64,64,64,64,4,60,4,0
 [563] 223: 24,36,0,0,0,0,0,0,0,36,24,0
 [WX3] 224: 56,0,68,68,68,68,68,68,68,0,56,0
 [M63] 225: 0,31,32,4,0,0,0,0,0,36,24,0
 [714] 226: 65,127,9,64,68,68,68,68,68,0,56,0
 [E63] 227: 24,36,0,0,0,0,0,4,32,31,0,0
 [B14] 228: 56,0,68,68,68,68,68,64,9,127,65,0
 [H83] 229: 0,0,60,16,0,32,0,0,0,48,16,0
 [CU3] 230: 68,68,124,4,36,0,64,64,64,0,32,0
 [F22] 231: 20,32,0,0,0,0,0,0,0,4,40,0
 [W44] 232: 36,20,84,84,84,84,84,84,84,80,72,0
 [FV2] 233: 0,0,0,0,252,0,0,0,0,4,0,0
 [VR3] 234: 0,32,32,32,248,36,36,36,4,0,0,0
 [7V2] 235: 0,56,4,0,0,0,0,0,4,56,0,0
 [YR3] 236: 64,120,64,4,4,4,4,4,64,124,68,0
 [SW2] 237: 0,48,8,0,4,0,4,0,8,48,0,0
 [2S3] 238: 64,112,64,8,0,4,0,8,64,112,64,0
 [2V2] 239: 0,60,0,0,4,8,4,0,0,60,0,0
 [YR3] 240: 64,120,68,4,0,8,0,4,68,120,64,0
 [H63] 241: 0,36,0,24,0,0,0,24,0,36,0,0
 [M14] 242: 68,68,108,0,16,16,16,0,108,68,68,0
 [223] 243: 0,49,12,0,0,1,0,6,8,48,0,0
 [UF3] 244: 66,98,89,5,5,4,6,4,88,96,64,0
 [N03] 245: 32,0,4,0,8,0,16,0,32,0,4,0
 [6C4] 246: 68,68,68,76,68,84,68,100,68,68,68,0
 [MC3] 247: 0,0,124,128,128,128,0,0,0,0,0,0
 [S63] 248: 32,32,216,4,4,4,0,0,0,0,0,0
 [VU2] 249: 0,0,0,0,0,236,0,0,0,0,0,0
 [EU2] 250: 0,0,0,0,0,220,0,0,0,0,0,0
 [MS3] 251: 0,0,0,0,0,0,128,128,128,124,0,0
 [473] 252: 0,0,0,0,0,0,4,4,4,216,32,32
 [1S3] 253: 64,0,128,128,0,64,64,0,128,0,0,0
 [YA3] 254: 0,128,0,0,128,0,0,128,0,0,0,0
 [0G3] 255: 12,16,0,32,0,64,0,32,0,16,12,0
 [HF3] 256: 28,4,36,4,68,4,68,4,36,4,28,0
 [693] 257: 0,0,48,0,72,72,72,76,0,0,0,0
 [263] 258: 0,0,32,84,4,4,12,0,80,0,0,0
 [0S3] 259: 128,184,4,0,0,0,0,0,4,184,128,0
 [YR3] 260: 64,120,64,4,4,4,4,64,124,68,0
 [RN3] 261: 24,36,0,0,0,0,0,0,128,164,144,0
 [254] 262: 56,16,84,84,84,84,84,84,16,40,0
 [NV4] 263: 12,32,128,128,128,128,128,128,36,24,0
 [SP4] 264: 168,148,84,84,84,84,84,84,144,188,0
 [8B3] 265: 140,32,0,0,0,0,0,0,36,152,0
 [B44] 266: 40,20,84,84,84,84,84,84,16,60,0
 [4R3] 267: 140,160,128,0,0,0,0,0,36,24,0
 [B44] 268: 40,20,84,84,84,84,84,84,16,60,0
 [0C3] 269: 12,32,0,0,0,192,0,0,36,24,0
 [694] 270: 40,20,84,84,84,212,84,84,16,60,0
 [Y73] 271: 0,0,16,32,8,8,8,12,32,0,0,0
 [MH3] 272: 0,0,32,20,68,68,76,64,16,0,0,0
 [AW4] 273: 24,36,128,128,128,128,128,128,36,16,0
 [PN4] 274: 184,144,84,84,84,84,84,84,144,168,0
 [4C3] 275: 152,36,0,0,0,0,0,0,36,144,0
 [254] 276: 56,16,84,84,84,84,84,84,16,40,0
 [2S3] 277: 152,164,128,0,0,0,0,0,36,16,0
 [254] 278: 56,16,84,84,84,84,84,84,16,40,0
 [373] 279: 0,0,0,128,0,60,128,0,0,0,0,0
 [U83] 280: 0,0,0,68,68,124,4,4,0,0,0,0
 [2P3] 281: 0,0,0,128,128,188,128,0,0,0,0,0
 [784] 282: 0,128,128,68,68,124,4,132,128,0,0,0
 [4R2] 283: 0,0,0,0,0,60,0,0,0,0,0,0
 [4S3] 284: 0,128,128,196,68,124,4,4,0,0,0,0
 [014] 285: 128,4,8,48,80,144,80,48,8,4,128,0
 [YV3] 286: 4,12,20,32,64,128,64,32,20,12,4,0
 [593] 287: 0,4,8,48,0,128,0,48,8,4,0,0
 [DX3] 288: 4,12,20,48,80,208,80,48,20,12,4,0
 [WN3] 289: 0,0,60,0,0,0,24,0,128,128,164,0
 [4C4] 290: 68,68,124,84,84,84,68,68,68,68,0
 [E53] 291: 12,32,0,0,0,0,60,0,0,0,36,0
 [0A4] 292: 40,20,84,84,84,84,124,84,84,84,0
 [0R4] 293: 28,16,48,16,80,16,252,144,144,128,0
 [2H3] 294: 28,32,0,64,0,128,252,4,4,4,0,0
 [5W4] 295: 24,36,128,128,128,128,128,128,36,24,0

[YP4] 296: 184,128,68,68,68,68,68,68,128,184,0
 [ZB3] 297: 152,36,0,0,0,0,0,0,0,36,152,0
 [WX3] 298: 56,0,68,68,68,68,68,68,68,0,56,0
 [ZR3] 299: 152,164,128,0,0,0,0,0,36,24,0
 [WX3] 300: 56,0,68,68,68,68,68,68,68,0,56,0
 [PR4] 301: 32,56,36,128,128,128,128,36,56,32,0
 [3A4] 302: 128,184,128,4,4,4,4,4,128,188,132,0
 [DB3] 303: 32,56,36,0,0,0,0,0,36,56,32,0
 [YL3] 304: 128,184,128,4,4,4,4,4,0,60,4,0
 [RK3] 305: 128,49,12,0,0,1,0,6,8,48,128,0
 [UF3] 306: 66,98,89,5,5,4,6,4,88,96,64,0
 [SR4] 307: 176,72,4,128,128,128,128,4,72,176,0
 [NT3] 308: 48,72,128,4,4,4,4,4,128,72,48,0
 [0S3] 309: 128,184,4,0,0,0,0,0,4,184,128,0
 [AR3] 310: 64,120,64,4,4,4,4,64,120,64,0
 [7B3] 311: 0,16,32,8,8,204,8,8,8,8,0,0
 [Z24] 312: 0,32,16,64,64,204,64,64,64,64,0
 [AV3] 313: 0,4,60,64,0,128,128,128,0,64,4,0
 [TW3] 314: 36,36,124,36,164,4,4,4,132,4,64,0
 [LB4] 315: 136,200,168,8,8,28,8,8,168,200,136,0
 [894] 316: 32,224,32,36,36,60,36,36,32,224,32,0
 [265] 317: 128,128,252,160,160,160,168,172,168,0,64,0
 [MA3] 318: 4,4,252,4,4,0,0,24,4,196,4,0
 [X14] 319: 0,2,0,16,18,124,16,144,128,64,64,0
 [UB3] 320: 4,4,2,2,0,124,128,0,0,128,0,0
 [4N3] 321: 12,32,0,0,0,0,0,0,128,164,152,0
 [B44] 322: 40,20,84,84,84,84,84,84,16,60,0
 [4B3] 323: 0,0,0,0,188,128,128,0,0,0,0
 [U83] 324: 0,0,0,68,68,124,4,4,0,0,0,0
 [MN3] 325: 24,36,0,0,0,0,0,0,128,164,152,0
 [WX3] 326: 56,0,68,68,68,68,68,68,68,0,56,0
 [PR3] 327: 0,56,4,0,0,0,0,128,132,184,128,0
 [YR3] 328: 64,120,64,4,4,4,4,64,124,68,0
 [6L5] 329: 128,156,160,128,128,128,128,160,156,128,0
 [LV3] 330: 68,124,4,64,64,64,64,64,4,60,4,0
 [SL5] 331: 128,188,160,128,144,128,136,128,132,188,128,0
 [784] 332: 68,124,68,100,0,16,0,76,68,124,68,0
 [7R4] 333: 112,0,136,136,136,136,136,248,8,8,0
 [4F3] 334: 100,148,4,4,4,4,4,4,244,4,4,0
 [LU4] 335: 112,0,136,136,136,136,136,136,136,112,0
 [GS3] 336: 100,148,4,4,4,4,4,4,148,100,0
 [N03] 337: 0,4,8,0,16,160,0,0,4,8,0,0
 [T03] 338: 0,8,4,20,4,100,4,4,0,8,0,0
 [0P2] 339: 12,0,0,0,0,0,0,0,0,0,0,0
 [5X3] 340: 28,16,16,16,16,16,16,16,16,16,0,0
 [2R2] 341: 0,0,0,0,0,0,0,0,0,12,0,0
 [2X3] 342: 16,16,16,16,16,16,16,16,16,28,0,0
 [NT3] 343: 0,68,100,8,8,16,16,16,36,32,64,0,8
 [724] 344: 0,36,232,40,16,16,32,36,92,92,28,4
 [KK3] 345: 0,4,100,8,8,16,20,36,44,76,0,0
 [8S3] 346: 0,4,200,8,16,16,32,40,64,92,0,0
 [KU2] 347: 0,0,0,0,0,222,0,0,0,0,0,0
 [0V2] 348: 0,0,0,0,0,188,0,0,0,0,0,0
 [283] 349: 0,16,0,40,0,84,0,40,0,68,0,0
 [PG3] 350: 0,0,48,0,72,0,180,0,72,0,132,0
 [W83] 351: 0,0,68,0,40,0,84,0,40,0,16,0
 [9G3] 352: 0,132,0,72,0,180,0,72,0,48,0,0
 [HC3] 353: 255,0,0,0,255,0,0,0,255,0,0,0
 [HC3] 354: 0,0,255,0,0,255,0,0,255,0,255,0
 [E84] 355: 0,255,0,255,0,255,0,255,0,255,0,255
 [E84] 356: 255,0,255,0,255,0,255,0,255,0,255,0
 [8U4] 357: 146,36,73,146,36,73,146,36,73,146,36,73
 [8U4] 358: 73,146,36,73,146,36,73,146,36,73,146,36
 [VU2] 359: 0,0,0,0,0,255,0,0,0,0,0,0
 [VU2] 360: 0,0,0,0,0,255,0,0,0,0,0,0
 [BY2] 361: 8,8,8,8,8,255,0,0,0,0,0,0
 [VU2] 362: 0,0,0,0,0,255,0,0,0,0,0,0
 [RF3] 363: 24,24,24,24,24,255,0,0,0,0,0,0
 [VU2] 364: 0,0,0,0,0,255,0,0,0,0,0,0
 [6S3] 365: 24,24,24,24,24,255,0,255,0,0,0,0
 [K63] 366: 0,0,0,0,0,255,0,255,0,0,0,0
 [SY2] 367: 8,8,8,8,8,15,8,15,0,0,0,0
 [GU2] 368: 0,0,0,0,0,15,0,15,0,0,0,0
 [9B3] 369: 24,24,24,24,24,31,0,0,0,0,0,0
 [TP2] 370: 0,0,0,0,0,31,0,0,0,0,0,0
 [6S3] 371: 24,24,24,24,24,255,0,255,0,0,0,0
 [S63] 372: 0,0,0,0,0,239,0,255,0,0,0,0
 [K63] 373: 0,0,0,0,0,255,0,255,0,0,0,0
 [K63] 374: 0,0,0,0,0,255,0,255,0,0,0,0
 [CP3] 375: 24,24,24,24,24,31,16,31,0,0,0,0
 [AU2] 376: 0,0,0,0,0,15,0,31,0,0,0,0
 [MS3] 377: 24,24,24,24,24,248,8,248,0,0,0,0
 [P53] 378: 0,0,0,0,0,224,0,240,0,0,0,0
 [GB3] 379: 8,8,8,8,8,248,8,248,0,0,0,0
 [G53] 380: 0,0,0,0,0,240,0,240,0,0,0,0
 [YF3] 381: 24,24,24,24,24,248,0,0,0,0,0,0
 [GU2] 382: 0,0,0,0,0,240,0,0,0,0,0,0
 [SU2] 383: 8,8,8,8,8,15,0,0,0,0,0,0
 [NP2] 384: 0,0,0,0,0,15,0,0,0,0,0,0
 [GZ2] 385: 0,0,0,0,0,248,8,8,8,8,8,8
 [GU2] 386: 0,0,0,0,0,240,0,0,0,0,0,0
 [033] 387: 8,8,8,8,8,248,8,8,8,8,8,8
 [GU2] 388: 0,0,0,0,0,240,0,0,0,0,0,0
 [5X2] 389: 8,8,8,8,8,15,8,8,8,8,8,8
 [NP2] 390: 0,0,0,0,0,15,0,0,0,0,0,0


```

[BZ2] 391: 0,0,0,0,0,255,8,8,8,8,8,8
[VU2] 392: 0,0,0,0,0,255,0,0,0,0,0,0
[MY2] 393: 8,8,8,8,8,8,8,8,8,8,8,8
[MN2] 394: 0,0,0,0,0,0,0,0,0,0,0,0
[V23] 395: 8,8,8,8,8,255,8,8,8,8,8,8
[VU2] 396: 0,0,0,0,0,255,0,0,0,0,0,0
[XP3] 397: 0,0,0,0,0,255,24,24,24,24,24,24
[VU2] 398: 0,0,0,0,0,255,0,0,0,0,0,0
[K93] 399: 0,0,0,0,0,255,0,255,8,8,8,8
[K63] 400: 0,0,0,0,0,255,0,255,0,0,0,0
[MS3] 401: 0,0,0,0,0,248,8,248,24,24,24,24
[U53] 402: 0,0,0,0,0,240,0,224,0,0,0,0
[XG3] 403: 0,0,0,0,0,31,16,31,24,24,24,24
[MU2] 404: 0,0,0,0,0,31,0,15,0,0,0,0
[S94] 405: 24,24,24,24,24,24,248,8,248,24,24,24,24
[G63] 406: 0,0,0,0,0,224,0,224,0,0,0,0
[G74] 407: 24,24,24,24,24,31,16,31,24,24,24,24
[GU2] 408: 0,0,0,0,0,15,0,15,0,0,0,0
[6R3] 409: 0,0,0,0,0,255,0,255,24,24,24,24
[W63] 410: 0,0,0,0,0,255,0,239,0,0,0,0
[G74] 411: 24,24,24,24,24,24,24,24,24,24,24,24
[MN2] 412: 0,0,0,0,0,0,0,0,0,0,0,0
[A94] 413: 24,24,24,24,24,24,255,0,255,24,24,24,24
[273] 414: 0,0,0,0,0,239,0,239,0,0,0,0
[B94] 415: 24,24,24,24,24,24,24,24,248,24,24,24,24
[PU2] 416: 0,0,0,0,0,0,0,0,224,0,0,0,0
[GE3] 417: 8,8,8,8,8,248,8,248,8,8,8,8
[G53] 418: 0,0,0,0,0,240,0,240,0,0,0,0
[B74] 419: 24,24,24,24,24,31,24,24,24,24,24,24
[NP2] 420: 0,0,0,0,0,15,0,0,0,0,0,0
[S13] 421: 8,8,8,8,8,15,8,15,8,8,8,8
[GU2] 422: 0,0,0,0,0,15,0,15,0,0,0,0
[GB3] 423: 0,0,0,0,0,248,8,248,8,8,8,8
[G53] 424: 0,0,0,0,0,240,0,240,0,0,0,0
[2R3] 425: 0,0,0,0,0,248,24,24,24,24,24,24
[GU2] 426: 0,0,0,0,0,240,0,0,0,0,0,0
[2G3] 427: 0,0,0,0,0,31,24,24,24,24,24,24
[TP2] 428: 0,0,0,0,0,31,0,0,0,0,0,0
[BY2] 429: 0,0,0,0,0,15,8,15,8,8,8,8
[GU2] 430: 0,0,0,0,0,15,0,15,0,0,0,0
[2E3] 431: 8,8,8,8,8,255,8,255,8,8,8,8
[K63] 432: 0,0,0,0,0,255,0,255,0,0,0,0
[X84] 433: 24,24,24,24,24,24,255,24,24,24,24,24
[VU2] 434: 0,0,0,0,0,255,0,0,0,0,0,0
[GY2] 435: 8,8,8,8,8,248,0,0,0,0,0,0
[GU2] 436: 0,0,0,0,0,240,0,0,0,0,0,0
[NT2] 437: 0,0,0,0,0,15,8,8,8,8,8,8
[NP2] 438: 0,0,0,0,0,15,0,0,0,0,0,0
[SZ5] 439: 255,255,255,255,255,255,255,255,255,255,255
[SZ5] 440: 255,255,255,255,255,255,255,255,255,255,255
[G74] 441: 15,15,15,15,15,15,15,15,15,15,15
[G74] 442: 15,15,15,15,15,15,15,15,15,15,15
[PA4] 443: 255,255,255,255,255,255,0,0,0,0,0
[PA4] 444: 255,255,255,255,255,255,0,0,0,0,0
[DB4] 445: 0,0,0,0,0,255,255,255,255,255,255
[DB4] 446: 0,0,0,0,0,255,255,255,255,255,255
[4R5] 447: 240,240,240,240,240,240,240,240,240,240,240
[WU5] 448: 224,224,224,224,224,224,224,224,224,224,224
[NV2] 449: 12,2,2,2,2,10,0,4,8,2,0,0
[HH3] 450: 12,16,16,16,16,0,12,8,4,16,0,0
[4R4] 451: 127,0,128,146,146,146,146,146,0,108,0,0
[PH3] 452: 127,128,0,0,0,0,0,180,72,0,0,0
[883] 453: 0,0,0,0,62,32,32,32,48,0,0,0
[TU2] 454: 0,0,0,0,60,0,0,0,32,0,0,0
[P73] 455: 16,16,16,0,14,0,8,0,14,0,0,0
[4D3] 456: 0,0,0,16,28,16,0,16,28,16,0,0
[385] 457: 130,198,130,170,130,146,130,130,130,130,0,0
[CB3] 458: 132,0,72,0,48,0,0,0,132,0,0,0
[GX3] 459: 28,0,34,34,34,34,34,32,60,32,32,0
[563] 460: 24,36,0,0,0,0,0,36,24,0,0,0
[7W2] 461: 28,3,2,2,2,2,2,0,28,0,0,0
[ZU2] 462: 25,6,0,0,0,0,0,4,24,0,0,0
[S53] 463: 0,0,16,16,0,14,0,16,0,0,0,0
[873] 464: 0,16,0,0,16,12,16,16,0,0,0,0
[VS4] 465: 0,130,146,170,130,198,130,170,146,130,0,0
[MD3] 466: 0,0,48,0,72,132,72,0,48,0,0,0
[2H3] 467: 0,0,16,56,16,84,16,56,16,0,0,0
[L73] 468: 0,0,48,0,72,0,72,0,48,0,0,0
[DW3] 469: 18,10,38,32,32,32,32,38,10,18,0,0
[DA3] 470: 16,32,12,0,0,0,0,12,32,16,0,0
[B23] 471: 4,8,66,2,34,2,18,2,8,4,0,0
[A14] 472: 8,4,16,208,144,176,144,144,4,8,0,0
[GT3] 473: 8,16,16,16,16,8,16,16,16,16,8,0
[743] 474: 16,8,8,8,8,16,8,8,8,8,16,0
[MT3] 475: 8,16,18,16,20,8,16,48,16,16,8,0
[ME3] 476: 16,8,8,12,8,16,40,8,72,8,16,0
[1W3] 477: 0,0,0,56,16,84,16,146,146,130,0,0
[G63] 478: 0,0,0,48,72,0,132,0,0,0,0,0
[8R4] 479: 126,0,128,128,128,128,128,0,126,0,0
[4S3] 480: 124,128,0,0,0,0,0,128,124,0,0,0
[FX3] 481: 18,18,18,18,18,18,18,18,18,18,0,0
[4X2] 482: 8,8,8,8,8,8,8,8,8,8,8,0
[LY3] 483: 18,18,18,18,18,58,18,18,18,18,0,0

```

```

[AU2] 484: 0,0,0,0,0,120,0,0,0,0,0,0
[DW2] 485: 2,2,2,2,2,42,2,18,2,2,0,0
[4W2] 486: 0,0,0,0,72,0,48,0,0,0,0,0
[WW2] 487: 2,2,2,2,18,2,42,2,2,2,0,0
[ZU2] 488: 0,0,0,0,0,48,0,72,0,0,0,0
[EC3] 489: 0,0,0,0,0,126,0,128,128,0,0,0
[PA3] 490: 0,0,0,0,0,124,128,0,0,128,0,0
[9V2] 491: 4,0,2,2,0,252,0,0,0,0,0,0
[SV2] 492: 0,4,0,0,4,248,0,0,0,0,0,0
[7V2] 493: 0,0,0,0,36,36,0,0,0,0,0,0
[8X3] 494: 16,16,16,16,84,84,16,16,16,16,0,0
[K83] 495: 4,4,4,32,32,4,4,4,32,32,0,0
[5D3] 496: 32,32,32,8,8,32,32,32,8,8,0,0
[BR3] 497: 0,0,0,64,128,128,128,64,0,0,0,0
[5H3] 498: 0,0,0,128,64,64,64,128,0,0,0,0
[DV2] 499: 0,0,0,0,16,16,0,0,0,0,0,0
[DV2] 500: 0,0,0,0,16,16,0,0,0,0,0,0
[MN2] 501: 0,0,0,0,0,0,0,0,0,0,0,0
[4R2] 502: 0,0,0,0,0,16,0,0,0,0,0,0
[TB3] 503: 0,0,0,0,8,0,4,0,2,254,128,128
[3D3] 504: 16,16,16,16,0,8,0,4,0,252,0,0
[EU3] 505: 224,128,128,128,96,0,0,0,0,0,0,0
[G63] 506: 224,0,0,0,224,0,0,0,0,0,0,0
[S63] 507: 0,160,128,64,0,0,0,0,0,0,0,0
[ZA3] 508: 160,32,96,160,0,0,0,0,0,0,0,0
[863] 509: 0,0,0,0,24,24,24,24,0,0,0,0
[K63] 510: 0,0,0,0,16,16,16,16,0,0,0,0
[MN2] 511: 0,0,0,0,0,0,0,0,0,0,0,0
[MN2] 512: 0,0,0,0,0,0,0,0,0,0,0,0

```

Listing 4. Die Zeichensatzdatei »roman.tbl«

```

[MN2] 1: 0,0,0,0,0,0,0,0,0,0,0,0
[MN2] 2: 0,0,0,0,0,0,0,0,0,0,0,0
[C65] 3: 126,1,128,160,162,138,138,162,160,128,1,126
[DG4] 4: 126,128,1,69,69,21,21,69,69,1,128,126
[UW5] 5: 126,127,255,223,223,245,245,223,223,255,127
126
[K06] 6: 126,254,255,187,187,235,235,187,187,255,254
126
[P34] 7: 16,16,56,56,28,28,28,56,56,16,16,0
[WW3] 8: 0,48,48,56,56,28,56,56,48,48,0,0
[2G3] 9: 0,0,24,24,60,60,60,24,24,0,0,0
[PX3] 10: 0,16,16,56,56,124,56,56,16,16,0,0
[PT3] 11: 0,0,40,40,120,124,120,40,40,0,0,0
[354] 12: 0,0,24,116,116,124,116,116,24,0,0,0
[CH3] 13: 0,0,24,26,58,62,58,26,24,0,0,0
[PX3] 14: 0,16,16,56,56,124,56,56,16,16,0,0
[CT2] 15: 0,0,0,8,8,28,8,8,0,0,0,0
[2Z2] 16: 0,0,0,0,24,24,24,0,0,0,0,0
[A06] 17: 255,255,255,247,247,227,227,247,247,255,255
255
[0X5] 18: 255,255,255,255,231,231,231,231,255,255,255
255
[S93] 19: 0,8,4,16,16,16,16,4,8,0,0,0
[RX2] 20: 0,8,16,4,4,4,4,16,8,0,0,0
[MW3] 21: 62,62,34,34,42,34,34,62,62,0,0,0
[EP3] 22: 60,60,36,0,24,0,36,60,60,60,0,0
[LT3] 23: 0,8,4,16,16,16,48,16,68,104,0,0
[RV3] 24: 0,8,16,4,4,36,132,196,144,200,0,0
[4S4] 25: 96,0,144,144,144,156,144,144,144,0,96,0
[XK3] 26: 64,160,0,8,8,28,8,8,0,160,64,0
[704] 27: 4,4,4,4,252,128,128,128,0,0,0,0
[7T3] 28: 0,12,12,12,248,64,64,64,64,0,0,0
[LH4] 29: 4,4,4,4,252,128,136,136,136,136,248,0
[194] 30: 0,12,12,12,248,64,64,88,88,240,0,0
[L74] 31: 32,40,32,48,48,112,48,48,32,40,32,0
[UW3] 32: 0,32,48,48,96,96,96,48,48,32,0,0
[PR3] 33: 124,124,56,56,16,16,0,0,0,0,0,0
[RP3] 34: 252,120,120,48,48,0,0,0,0,0,0,0
[KP3] 35: 0,0,0,0,16,16,56,56,124,124,0,0
[HN3] 36: 0,0,0,0,0,48,48,120,120,252,0,0
[ZD3] 37: 0,0,0,64,68,252,68,64,0,0,0,0
[5E3] 38: 0,0,0,8,136,252,136,8,0,0,0,0
[863] 39: 0,0,0,244,0,0,244,0,0,0,0,0
[863] 40: 0,0,0,244,0,0,244,0,0,0,0,0
[ZN4] 41: 0,0,64,32,128,128,128,252,128,128,252,0
[HW3] 42: 0,0,64,128,32,32,32,252,0,0,252,0
[GL4] 43: 0,68,32,144,160,160,160,160,156,0,64,0
[R04] 44: 8,64,132,36,20,20,20,20,32,136,0,0
[MT2] 45: 4,4,4,4,4,4,4,4,4,4,4,0
[MT2] 46: 4,4,4,4,4,4,4,4,4,4,4,0

```

Listing 5. Die Zeichensatzdatei »double.tbl«

[TC3] 47: 0,0,0,72,72,252,72,72,0,0,0,0
 [LE3] 48: 4,4,4,4,140,252,140,4,4,4,4
 [3D3] 49: 0,0,0,64,64,252,64,64,0,0,0,0
 [ZC3] 50: 0,0,0,0,128,252,128,0,0,0,0
 [DV2] 51: 0,0,0,0,4,252,4,0,0,0,0,0
 [NX2] 52: 0,0,0,8,8,252,8,8,0,0,0,0
 [SU2] 53: 0,0,0,0,0,0,36,0,24,0,0
 [V14] 54: 16,16,16,16,16,16,16,16,56,16,16,0
 [RU2] 55: 0,24,0,36,0,0,0,0,0,0,0,0
 [324] 56: 16,16,56,16,16,16,16,16,16,16,0
 [MP2] 57: 56,0,0,0,0,0,0,0,0,0,0,0
 [MV2] 58: 56,8,8,8,8,8,8,8,8,8,0,0
 [EB3] 59: 0,24,24,60,0,0,0,0,60,24,24,0
 [334] 60: 16,16,56,56,16,16,16,56,16,16,0
 [9E4] 61: 16,16,48,48,112,112,112,48,16,16,0
 [RR4] 62: 16,48,48,112,112,240,112,112,48,16,0
 [DH3] 63: 8,8,12,12,14,14,14,12,12,8,8,0
 [524] 64: 16,24,24,28,28,30,28,24,24,16,0
 [MN2] 65: 0,0,0,0,0,0,0,0,0,0,0,0
 [MN2] 66: 0,0,0,0,0,0,0,0,0,0,0,0
 [263] 67: 0,0,0,0,250,250,0,0,0,0,0,0
 [W63] 68: 0,0,0,0,244,244,0,0,0,0,0,0
 [FA3] 69: 0,0,0,192,192,0,0,192,0,0,0,0
 [YA3] 70: 0,0,0,128,128,0,0,128,0,0,0,0
 [884] 71: 32,32,254,254,32,32,254,32,32,0,0
 [P84] 72: 16,16,252,252,16,16,252,16,16,0,0
 [8C4] 73: 32,84,84,254,254,84,84,254,84,8,0
 [CN3] 74: 96,0,0,252,252,0,0,252,0,0,24,0
 [EV3] 75: 4,68,72,8,16,16,32,32,68,196,0,0
 [Y74] 76: 4,140,136,16,16,32,32,64,200,136,0,0
 [P64] 77: 68,222,186,162,130,210,0,12,0,22,0,0
 [KW3] 78: 140,220,112,0,96,0,148,0,24,4,0,0
 [V63] 79: 0,0,0,0,32,192,192,0,0,0,0,0
 [L53] 80: 0,0,0,0,0,192,128,0,0,0,0,0
 [ZE3] 81: 56,124,198,130,0,0,0,0,0,0,0,0
 [AB3] 82: 120,252,132,0,0,0,0,0,0,0,0,0
 [SH3] 83: 0,0,0,0,0,130,198,124,56,0,0,0
 [Y93] 84: 0,0,0,0,0,0,132,252,120,0,0,0
 [2S3] 85: 0,12,12,24,24,56,24,24,12,12,0,0
 [M14] 86: 20,16,24,24,56,48,56,24,24,16,20,0
 [8A4] 87: 16,16,16,16,124,124,16,16,16,16,0
 [C53] 88: 0,0,0,0,120,120,0,0,0,0,0,0
 [WP2] 89: 0,0,0,1,6,6,0,0,0,0,0,0
 [GP2] 90: 0,0,0,0,6,4,0,0,0,0,0,0
 [L14] 91: 16,16,16,16,16,16,16,16,16,16,0
 [MN2] 92: 0,0,0,0,0,0,0,0,0,0,0,0
 [MP2] 93: 0,0,0,0,4,4,4,0,0,0,0,0
 [MP2] 94: 0,0,0,0,4,4,4,0,0,0,0,0
 [GA3] 95: 0,4,4,8,8,16,16,32,32,64,0,0
 [YD3] 96: 0,4,8,8,16,16,32,32,64,64,0,0
 [3G5] 97: 124,124,130,130,130,130,130,64,124,0,0
 [2P3] 98: 120,252,0,0,0,0,0,132,120,0,0
 [XD3] 99: 0,0,34,2,66,126,254,2,2,2,0,0
 [W63] 100: 0,0,0,64,0,252,252,0,0,0,0,0
 [GV4] 101: 14,142,146,146,146,146,146,226,98,0,0
 [1F3] 102: 140,28,0,0,0,0,0,224,192,0,0
 [ZA5] 103: 130,130,130,146,146,146,146,108,108,0,0
 [3G3] 104: 0,0,0,0,0,0,0,252,72,0,0
 [NF3] 105: 16,0,32,0,64,0,252,252,0,0,0,0
 [H74] 106: 16,48,16,80,20,148,252,252,20,0,0
 [PB5] 107: 226,242,146,146,146,146,146,140,140,0,0
 [S83] 108: 224,224,0,0,0,0,0,28,8,0,0,0
 [S55] 109: 124,124,146,146,146,146,146,128,12,0,0
 [F73] 110: 120,252,0,0,0,0,0,20,8,0,0,0
 [MC5] 111: 128,128,128,128,134,142,152,176,224,192,0,0
 [FG3] 112: 0,0,0,0,4,12,24,48,224,192,0,0
 [Q15] 113: 108,108,146,146,146,146,146,108,108,0,0
 [U93] 114: 72,252,0,0,0,0,0,180,72,0,0,0
 [ZT4] 115: 96,2,146,146,146,146,146,252,124,0,0
 [TE3] 116: 64,160,0,0,0,0,0,252,248,0,0
 [P03] 117: 0,0,0,0,68,68,68,0,0,0,0,0
 [XC3] 118: 0,0,0,0,136,136,136,0,0,0,0,0
 [Q03] 119: 0,0,0,0,70,68,68,0,0,0,0,0
 [VB3] 120: 0,0,0,2,136,140,140,0,0,0,0,0
 [3D3] 121: 0,16,16,40,40,68,68,0,0,0,0,0
 [9S3] 122: 0,0,48,48,72,72,132,132,0,0,0,0
 [MN2] 123: 0,0,0,0,0,0,0,0,0,0,0,0
 [8Z3] 124: 40,40,40,40,40,40,40,40,40,40,0
 [3D3] 125: 0,0,0,0,68,68,68,40,40,16,16,0
 [XR3] 126: 0,0,0,0,132,132,72,72,48,48,0,0
 [E94] 127: 0,64,0,128,128,138,128,144,96,96,0,0
 [UB3] 128: 0,0,128,0,0,8,16,0,224,64,0,0
 [VB5] 129: 120,124,128,128,176,136,184,136,112,0,0
 [AU3] 130: 120,248,4,4,52,68,52,4,4,244,0,0
 [Z84] 131: 62,62,0,16,144,144,16,80,16,62,0,0
 [5R3] 132: 60,124,0,128,0,0,128,0,64,60,0,0
 [215] 133: 254,254,146,146,146,146,146,108,0,0,0
 [QH3] 134: 252,252,0,0,0,0,0,180,72,0,0,0
 [9V4] 135: 124,124,130,130,130,130,130,130,0,0,0
 [UA3] 136: 120,252,0,0,0,0,0,132,0,0,0,0
 [MS4] 137: 254,254,130,130,130,130,68,56,0,0,0
 [5P3] 138: 252,252,0,0,0,0,0,132,0,120,0,0
 [6A5] 139: 254,254,146,146,146,146,130,130,130,0,0
 [K63] 140: 252,252,0,0,0,0,0,0,0,0,0,0
 [UC5] 141: 254,254,144,144,144,128,128,128,0,0,0
 [K63] 142: 252,252,0,0,0,0,0,0,0,0,0,0
 [335] 143: 124,124,130,130,130,146,146,146,30,0,0

[4B3] 144: 120,252,0,0,0,0,0,0,156,0,0,0
 [9A4] 145: 254,254,16,16,16,16,16,16,254,0,0
 [LB3] 146: 252,252,0,0,0,0,0,0,252,0,0
 [273] 147: 0,0,0,0,254,254,0,0,0,0,0,0
 [K63] 148: 0,0,0,0,252,252,0,0,0,0,0,0
 [Y73] 149: 4,0,2,2,2,2,2,2,252,252,0,0
 [R73] 150: 0,4,0,0,0,0,0,0,252,248,0,0
 [8X3] 151: 254,254,16,16,0,40,0,68,0,130,0,0
 [KR3] 152: 252,252,0,0,48,0,72,0,132,0,0,0
 [K83] 153: 254,254,2,2,2,2,2,2,2,0,0,0
 [K63] 154: 252,252,0,0,0,0,0,0,0,0,0,0
 [WN3] 155: 254,254,64,0,32,0,32,0,64,254,0,0
 [1T3] 156: 252,252,0,64,0,32,0,64,0,252,0,0
 [8S3] 157: 254,254,0,32,0,16,0,8,0,254,0,0
 [5T3] 158: 252,252,64,0,32,0,16,0,8,252,0,0
 [4V4] 159: 124,124,130,130,130,130,130,124,0,0,0
 [2P3] 160: 120,252,0,0,0,0,0,132,120,0,0,0
 [QX4] 161: 254,254,144,144,144,144,144,96,0,0,0
 [YG3] 162: 252,252,0,0,0,0,0,160,64,0,0,0
 [SW4] 163: 124,124,130,130,130,138,130,134,0,126,0,0
 [KP3] 164: 120,252,0,0,0,0,0,8,0,132,120,0,0
 [DX4] 165: 254,254,144,144,144,152,144,148,0,98,0,0
 [L63] 166: 252,252,0,16,0,8,0,164,64,0,0,0
 [VR4] 167: 96,98,146,146,146,146,146,146,140,12,0,0
 [RA3] 168: 68,224,0,0,0,0,0,28,136,0,0,0
 [2E5] 169: 128,128,128,128,254,254,128,128,128,0,0,0
 [K63] 170: 0,0,0,0,252,252,0,0,0,0,0,0
 [RC3] 171: 252,252,2,2,2,2,2,2,2,252,0,0
 [VC3] 172: 248,252,0,0,0,0,0,4,248,0,0,0
 [LT3] 173: 224,240,24,12,6,2,4,8,16,224,0,0
 [7T3] 174: 224,240,24,12,4,8,16,224,0,0,0
 [7E3] 175: 254,254,0,4,0,8,0,4,0,254,0,0
 [3D3] 176: 252,252,4,0,8,0,8,0,4,252,0,0
 [A34] 177: 198,198,40,40,16,16,0,40,0,198,0,0
 [S04] 178: 132,204,72,48,48,0,48,0,72,132,0,0
 [B84] 179: 192,224,32,16,30,30,16,0,32,192,0,0
 [Y14] 180: 192,192,32,32,28,28,0,32,0,192,0,0
 [1C5] 181: 134,142,138,146,146,162,162,194,194,130,0,0
 [284] 182: 12,12,16,16,32,32,64,64,128,128,0,0
 [Y64] 183: 254,254,130,130,130,130,0,0,0,0,0,0
 [K63] 184: 252,252,0,0,0,0,0,0,0,0,0,0
 [YT3] 185: 128,64,64,32,32,16,16,8,4,4,0,0
 [PZ3] 186: 128,128,64,64,32,32,16,16,8,4,4,0
 [Y64] 187: 0,0,0,0,130,130,130,130,254,254,0,0
 [K63] 188: 0,0,0,0,0,0,0,252,252,0,0,0
 [C93] 189: 0,8,8,16,16,32,0,16,0,8,0,0
 [3C3] 190: 0,0,16,16,32,32,32,0,16,0,0,0
 [MN2] 191: 0,0,0,0,0,0,0,0,0,0,0,0
 [SP2] 192: 1,1,1,1,1,1,1,1,1,1,1,1
 [E83] 193: 0,0,0,0,192,192,32,0,0,0,0,0
 [973] 194: 0,0,0,0,128,192,0,0,0,0,0,0
 [TS3] 195: 4,46,42,42,42,42,42,42,10,28,0,0
 [D63] 196: 12,12,0,0,0,0,0,0,32,28,0,0
 [D24] 197: 254,254,34,34,34,34,34,34,28,0,0
 [OC3] 198: 252,252,0,0,0,0,0,36,24,0,0,0
 [AX3] 199: 28,28,34,34,34,34,34,34,34,0,0,0
 [PU2] 200: 24,60,0,0,0,0,0,0,0,0,0,0
 [VB3] 201: 12,16,2,2,2,2,2,2,2,254,254,0,0
 [VZ3] 202: 24,4,32,32,32,32,32,32,252,252,0,0
 [FW3] 203: 28,60,42,42,42,42,42,42,42,16,0,0
 [A03] 204: 56,60,0,0,0,0,0,0,48,0,0,0
 [L54] 205: 0,16,16,126,126,144,144,128,0,0,0
 [NB3] 206: 0,0,0,0,124,252,0,0,128,0,0,0
 [WT3] 207: 28,0,34,34,34,34,34,34,63,63,0,0
 [M73] 208: 24,37,1,1,1,1,1,1,1,63,62,0,0
 [BC3] 209: 254,254,0,0,0,0,0,16,14,0,0,0
 [Z04] 210: 252,252,32,32,32,32,32,32,28,0,0,0
 [WU2] 211: 0,0,0,0,62,62,0,0,0,0,0,0
 [G83] 212: 0,0,0,0,188,188,0,0,0,0,0,0
 [C53] 213: 0,1,0,0,0,0,191,190,0,0,0,0
 [EW2] 214: 2,0,1,1,1,1,1,62,62,0,0,0
 [KK3] 215: 254,254,8,0,16,8,32,4,64,2,0,0,0
 [HS3] 216: 252,252,0,16,16,32,8,64,4,0,0,0
 [UN3] 217: 0,0,0,128,128,254,254,0,0,0,0,0
 [E53] 218: 0,0,0,0,252,252,0,0,0,0,0,0
 [VN3] 219: 62,62,32,0,30,30,32,32,0,30,0,0
 [DG3] 220: 60,60,0,32,28,60,0,0,32,28,0,0,0
 [YR3] 221: 62,30,32,32,32,32,32,32,0,30,0,0
 [K63] 222: 60,60,0,0,0,0,0,0,32,28,0,0,0
 [TT3] 223: 28,28,34,34,34,34,34,34,28,0,0,0
 [A63] 224: 24,60,0,0,0,0,0,0,36,24,0,0,0
 [CT3] 225: 63,63,34,34,34,34,34,34,28,0,0,0
 [N63] 226: 63,63,0,0,0,0,0,0,36,24,0,0,0
 [WT3] 227: 28,0,34,34,34,34,34,34,63,63,0,0,0
 [473] 228: 24,36,0,0,0,0,0,0,63,63,0,0,0
 [FP3] 229: 62,62,16,0,32,32,32,32,0,16,0,0
 [T43] 230: 60,60,0,32,0,0,0,0,32,0,0,0
 [AX3] 231: 18,58,42,42,42,42,42,42,46,36,0,0
 [H63] 232: 48,48,0,0,0,0,0,0,12,12,0,0,0
 [A63] 233: 0,0,0,252,254,2,2,2,2,0,0,0,0
 [CR3] 234: 0,32,32,252,252,32,32,0,4,0,0,0
 [A03] 235: 60,60,2,2,2,2,2,2,0,60,0,0,0
 [P03] 236: 56,60,0,0,0,0,0,0,4,56,0,0,0
 [X13] 237: 56,56,4,4,2,0,4,0,8,48,0,0,0
 [143] 238: 48,56,8,4,4,0,4,0,8,48,0,0,0
 [U13] 239: 62,60,4,0,8,8,0,8,0,62,0,0,0
 [613] 240: 60,60,0,8,0,8,0,8,0,4,60,0,0

So logisch und einfach kann Textverarbeitung sein!

Was Context PC nicht kann:

- Dokumente mit 4000 Seiten verwalten
- 3-fach geschachtelte Fußnoten
- Grafik aus Fremdprogrammen einbinden.

Wenn Sie diese Features unbedingt benötigen, empfehlen wir Ihnen die Textverarbeitung der Firma XXX!

Was CONTEXT PC dafür jedoch ausgezeichnet beherrscht:

- exakte Wiedergabe des Druckbildes auf dem Bildschirm
- automatische Worttrennung durch Trennhilfe mit 99% Treffer-Wahrscheinlichkeit.
- einfachste Bedienerlogik, Hilfsmenues für jede denkbare Situation
- integrierter Dateimanager
- Wordstar-Kompatibilität, über Tastaturmakros auch selbst konfigurierbar.
- Rechnen im Text, automatische Datum- und Zeiteinfügung, Kalender
- Grafik-Zeichen direkt aus Tabelle in Text einsetzen
- 5 parallel bearbeitbare Textpuffer, Textlänge nur durch eigene Hardware-Konfiguration begrenzt
- zweispaltiges Layout möglich.
- automatische BackUp-Funktion
- DOS-Aufrufe (auch andere Programme) aus Context heraus
- kontrollierbar über Kommandozeilen-Argumente
- hervorragend als Programm-Editor
- an jedem handelsüblichen Drucker einsetzbar

```

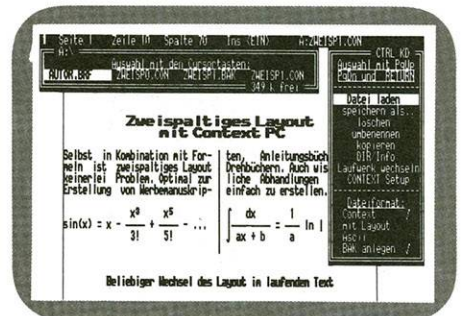
[2C3] 241: 34,34,20,20,8,8,0,20,0,34,0,0
[UC3] 242: 0,36,36,24,24,0,24,0,36,0,0,0
[V03] 243: 60,60,2,2,2,2,2,2,2,63,0,0
[413] 244: 56,61,5,1,1,1,1,1,1,62,0,0
[0X3] 245: 34,34,38,38,42,42,50,50,34,34,0,0
[CB3] 246: 0,4,4,8,8,16,16,32,32,0,0,0
[8U3] 247: 16,16,124,238,130,130,0,0,0,0,0
[K63] 248: 0,0,252,252,0,0,0,0,0,0,0,0
[ZU2] 249: 0,0,0,0,0,238,0,0,0,0,0,0
[MU2] 250: 0,0,0,0,0,204,0,0,0,0,0,0
[7S3] 251: 0,0,0,0,128,128,252,124,0,0,0,0
[DC3] 252: 0,0,0,0,4,4,220,248,32,32,0,0
[MW3] 253: 64,64,128,128,0,64,64,0,128,0,0,0
[GR3] 254: 0,128,128,0,128,0,0,128,0,0,0,0
[AS3] 255: 12,28,16,32,32,64,0,32,0,16,12,0
[6W3] 256: 28,28,36,36,68,68,4,36,4,28,0,0
[FD3] 257: 0,0,48,50,74,74,74,76,0,0,0,0
[5S3] 258: 0,0,32,112,0,0,8,4,80,0,0,0
[D93] 259: 188,60,2,2,2,2,2,2,2,0,188,0
[P03] 260: 56,60,0,0,0,0,0,0,4,56,0,0
[FW3] 261: 28,60,42,42,42,42,42,42,16,0,0,0
[TS3] 262: 56,60,0,0,0,0,0,128,128,176,0,0
[7L4] 263: 4,46,170,170,170,170,170,170,10,28,0,0
[XP3] 264: 140,140,0,0,0,0,0,0,160,156,0,0
[M84] 265: 132,46,42,42,42,42,42,42,158,0,0
[M22] 266: 44,44,0,0,0,0,0,0,0,60,0,0
[494] 267: 132,174,170,42,42,42,42,42,10,28,0,0
[D63] 268: 12,12,0,0,0,0,0,0,32,28,0,0
[F04] 269: 4,46,42,42,170,170,42,42,10,28,0,0
[HP3] 270: 12,12,0,0,128,128,0,0,32,28,0,0
[K73] 271: 0,48,56,8,10,10,4,0,0,0,0,0
[NU3] 272: 0,48,112,64,64,64,76,64,64,0,0,0
[5N4] 273: 28,60,170,170,170,170,170,170,42,16,0,0
[ZT3] 274: 184,188,0,0,0,0,0,0,128,176,0,0
[584] 275: 156,40,42,42,42,42,42,42,144,0,0
[MU2] 276: 56,4,0,0,0,0,0,0,0,48,0,0
[EA4] 277: 156,188,170,42,42,42,42,42,16,0,0
[A03] 278: 56,60,0,0,0,0,0,0,0,48,0,0
[WU2] 279: 0,0,0,0,62,62,0,0,0,0,0,0
[PB3] 280: 0,0,0,128,60,60,0,128,0,0,0,0
[PC3] 281: 0,0,0,0,190,190,128,0,0,0,0,0
[2Y3] 282: 0,0,128,128,60,60,0,128,128,0,0,0
[WU2] 283: 0,0,0,0,62,62,0,0,0,0,0,0
[8U3] 284: 0,0,128,128,188,0,0,0,0,0,0
[8S4] 285: 190,190,80,16,144,144,16,80,144,190,0,0
[5R3] 286: 60,124,0,128,0,0,128,0,64,60,0,0
[8W3] 287: 14,30,8,40,136,136,40,8,24,14,0,0
[TG3] 288: 28,28,32,0,64,64,0,32,0,28,0,0
[554] 289: 10,62,62,0,0,0,0,2,130,130,0,0
[Y84] 290: 64,124,124,64,64,64,64,64,64,64,0,0
[8T3] 291: 4,46,42,42,42,42,42,42,42,42,0,0
[122] 292: 12,12,0,0,0,0,60,0,0,0,0,0
[DP4] 293: 30,30,48,48,0,0,0,254,146,146,130,0
[HS3] 294: 28,60,32,64,64,128,252,0,0,0,0,0
[EP4] 295: 28,28,162,162,162,162,162,162,0,28,0,0
[MS3] 296: 152,188,0,0,0,0,0,0,164,152,0,0
[V14] 297: 156,28,34,34,34,34,34,34,0,156,0,0
[A63] 298: 24,60,0,0,0,0,0,0,36,24,0,0
[0A4] 299: 156,156,162,34,34,34,34,34,0,28,0,0
[A63] 300: 24,60,0,0,0,0,0,0,36,24,0,0
[SC4] 301: 60,60,130,130,130,130,130,130,0,60,0,0
[5T3] 302: 184,188,0,0,0,0,0,0,132,184,0,0
[A03] 303: 60,60,2,2,2,2,2,2,0,60,0,0
[ML3] 304: 184,188,128,0,0,0,0,0,4,56,0,0
[C93] 305: 188,60,2,2,2,2,2,2,191,0,0
[R03] 306: 57,61,1,1,1,1,1,1,1,62,0,0
[8U3] 307: 156,188,2,2,2,2,2,2,160,156,0,0
[KV3] 308: 56,60,64,64,64,64,64,64,4,56,0,0
[YV3] 309: 188,188,2,2,2,2,2,2,128,188,0,0
[UA3] 310: 120,124,0,0,0,0,0,0,4,120,0,0
[883] 311: 56,56,40,0,0,198,0,0,0,0,0,0
[334] 312: 48,48,72,72,72,204,72,72,72,72,0,0
[CV3] 313: 2,2,62,126,2,130,130,130,2,66,0,0
[KY3] 314: 32,36,124,124,160,32,0,0,128,0,0,0
[T94] 315: 208,240,48,30,30,16,16,16,48,208,0,0
[6V3] 316: 200,200,40,60,28,8,8,40,8,200,0,0
[015] 317: 254,254,128,128,128,128,140,34,66,0,0
[S84] 318: 252,252,32,32,32,40,40,44,136,72,0,0
[1X3] 319: 0,0,1,17,30,126,112,144,128,0,0,0
[YE3] 320: 0,2,0,0,30,124,224,0,0,128,0,0
[H94] 321: 4,46,42,42,42,42,42,42,170,138,156,0,0
[D63] 322: 12,12,0,0,0,0,0,0,32,28,0,0
[WU2] 323: 0,0,0,0,62,62,0,0,0,0,0,0
[MN3] 324: 0,0,0,0,60,60,128,128,128,0,0,0
[X94] 325: 28,28,34,34,34,34,34,162,128,156,0,0
[A63] 326: 24,60,0,0,0,0,0,0,36,24,0,0
[RS3] 327: 60,60,2,2,2,2,2,2,130,128,188,0,0
[P03] 328: 56,60,0,0,0,0,0,0,4,56,0,0
[835] 329: 190,158,160,160,160,160,160,160,128,30,0,0
[K63] 330: 60,60,0,0,0,0,0,0,32,28,0,0

```

Listing 5. Die Zeichensatzdatei »double.tbl«



Der Dateimanager des Context PC macht DOS-Benutzeroberflächen überflüssig.



»WYSIWYG« in Reinform: Alle Schriftattribute werden am Bildschirm dargestellt.



Notwendige Konfiguration:

PC XT/AT mit 256 kByte oder mehr, 1 Diskettenlaufwerk bzw. Festplatte, MS-DOS ab Version 2.0. CGA-, Hercules- und EGA-Grafikkarten werden unterstützt. Zum Lieferumfang gehört ein umfangreiches deutsches Handbuch inklusive Tutorial.

© 1988 DMV GmbH

CONTEXT PC

99,- DM (unverbindliche Preisempfehlung)

Wenn Sie über den DMV-Bestellservice bestellen, gilt folgendes:

Inland:		Ausland:	
Einzelpreis	99,- DM	Einzelpreis	99,- DM
zzgl. Versandkosten	3,- DM	zzgl. Versandkosten	5,- DM
Endpreis	102,- DM	Endpreis	104,- DM

Best.-Nr. 228 5 1/4" Disk. Best.-Nr. 234 3 1/2" Disk.

DMV GmbH

Abt. Software · Postfach 250 · 3440 Eschwege

– Bitte Bestellkarte benutzen –


```

[7C5] 331: 190,190,128,144,128,136,128,132,128,190,0,0
[8G3] 332: 124,60,32,0,16,0,8,0,4,124,0,0
[V73] 333: 18,58,2,2,2,34,26,2,2,2,0,0
[EG3] 334: 48,48,72,72,72,8,56,8,8,0,0
[M83] 335: 18,58,2,2,2,2,2,42,18,0,0
[DV3] 336: 48,48,72,72,72,72,72,0,48,0,0
[Y73] 337: 4,4,10,10,18,178,2,2,0,4,0,0
[S73] 338: 0,12,8,16,16,96,0,0,4,0,0,0
[0P2] 339: 12,0,0,0,0,0,0,0,0,0,0
[5X3] 340: 28,16,16,16,16,16,16,16,16,0,0
[2R2] 341: 0,0,0,0,0,0,0,0,12,0,0
[2X3] 342: 16,16,16,16,16,16,16,16,28,0,0
[NT3] 343: 0,68,100,8,8,16,16,36,32,64,0,8
[724] 344: 0,36,232,40,16,16,32,36,92,92,28,4
[KK3] 345: 0,4,100,8,8,16,20,36,44,76,0,0
[8S3] 346: 0,4,200,8,16,16,32,40,64,92,0,0
[KU2] 347: 0,0,0,0,0,222,0,0,0,0,0,0
[0V2] 348: 0,0,0,0,0,188,0,0,0,0,0,0
[283] 349: 0,16,0,40,0,84,0,40,0,68,0,0
[PG3] 350: 0,0,48,0,72,0,180,0,72,0,132,0
[W83] 351: 0,0,68,0,40,0,84,0,40,0,16,0
[9G3] 352: 0,132,0,72,0,180,0,72,0,48,0,0
[HC3] 353: 255,0,0,0,255,0,0,255,0,0,0
[HC3] 354: 0,0,255,0,0,255,0,0,255,0,0
[E84] 355: 0,255,0,255,0,255,0,255,0,255,0
[E84] 356: 255,0,255,0,255,0,255,0,255,0
[8U4] 357: 146,36,73,146,36,73,146,36,73,146,36,73
[8U4] 358: 73,146,36,73,146,36,73,146,36,73,146,36
[VU2] 359: 0,0,0,0,0,255,0,0,0,0,0,0
[VU2] 360: 0,0,0,0,0,255,0,0,0,0,0,0
[BY2] 361: 8,8,8,8,255,0,0,0,0,0,0
[VU2] 362: 0,0,0,0,0,255,0,0,0,0,0,0
[RF3] 363: 24,24,24,24,24,255,0,0,0,0,0
[VU2] 364: 0,0,0,0,0,255,0,0,0,0,0,0
[6S3] 365: 24,24,24,24,24,255,0,255,0,0,0
[K63] 366: 0,0,0,0,0,255,0,255,0,0,0
[SY2] 367: 8,8,8,8,15,8,15,0,0,0,0
[GU2] 368: 0,0,0,0,0,15,0,15,0,0,0
[9B3] 369: 24,24,24,24,24,31,0,0,0,0,0
[TP2] 370: 0,0,0,0,0,31,0,0,0,0,0
[6S3] 371: 24,24,24,24,24,255,0,255,0,0,0
[S63] 372: 0,0,0,0,0,239,0,255,0,0,0
[K63] 373: 0,0,0,0,0,255,0,255,0,0,0
[K63] 374: 0,0,0,0,0,255,0,255,0,0,0
[CP3] 375: 24,24,24,24,24,31,16,31,0,0,0
[AU2] 376: 0,0,0,0,0,15,0,31,0,0,0
[MS3] 377: 24,24,24,24,24,248,8,248,0,0,0
[P53] 378: 0,0,0,0,0,224,0,240,0,0,0
[GB3] 379: 8,8,8,8,248,8,248,0,0,0,0
[G53] 380: 0,0,0,0,0,240,0,240,0,0,0
[YF3] 381: 24,24,24,24,24,248,0,0,0,0,0
[GU2] 382: 0,0,0,0,0,240,0,0,0,0,0
[5U2] 383: 8,8,8,8,15,0,0,0,0,0,0
[NF2] 384: 0,0,0,0,0,15,0,0,0,0,0
[G22] 385: 0,0,0,0,0,248,8,8,8,8,8
[GU2] 386: 0,0,0,0,0,240,0,0,0,0,0
[033] 387: 8,8,8,8,248,8,8,8,8,8
[GU2] 388: 0,0,0,0,0,240,0,0,0,0,0
[5X2] 389: 8,8,8,8,15,8,8,8,8,8
[NF2] 390: 0,0,0,0,0,15,0,0,0,0,0
[BZ2] 391: 0,0,0,0,0,255,8,8,8,8,8
[VU2] 392: 0,0,0,0,0,255,0,0,0,0,0
[MY2] 393: 8,8,8,8,8,8,8,8,8,8,8
[MN2] 394: 0,0,0,0,0,0,0,0,0,0,0
[V23] 395: 8,8,8,8,8,255,8,8,8,8,8
[VU2] 396: 0,0,0,0,0,255,0,0,0,0,0
[XP3] 397: 0,0,0,0,0,255,24,24,24,24,24
[VU2] 398: 0,0,0,0,0,255,0,0,0,0,0
[K93] 399: 0,0,0,0,0,255,0,255,8,8,8
[K63] 400: 0,0,0,0,0,255,0,255,0,0,0
[MS3] 401: 0,0,0,0,0,248,8,248,24,24,24
[U53] 402: 0,0,0,0,0,240,0,224,0,0,0
[XG3] 403: 0,0,0,0,0,31,16,31,24,24,24
[MU2] 404: 0,0,0,0,0,31,0,15,0,0,0
[S94] 405: 24,24,24,24,24,248,8,248,24,24,24
[063] 406: 0,0,0,0,0,224,0,224,0,0,0
[074] 407: 24,24,24,24,24,31,16,31,24,24,24
[GU2] 408: 0,0,0,0,0,15,0,15,0,0,0
[6R3] 409: 0,0,0,0,0,255,0,255,24,24,24
[W63] 410: 0,0,0,0,0,255,0,239,0,0,0
[G74] 411: 24,24,24,24,24,24,24,24,24,24,24
[MN2] 412: 0,0,0,0,0,0,0,0,0,0,0
[A94] 413: 24,24,24,24,24,255,0,255,24,24,24
[273] 414: 0,0,0,0,0,239,0,239,0,0,0
[894] 415: 24,24,24,24,24,24,24,24,24,24,24
[P02] 416: 0,0,0,0,0,0,0,224,0,0,0
[GE3] 417: 8,8,8,8,8,248,8,248,8,8,8
[G53] 418: 0,0,0,0,0,240,0,240,0,0,0
[B74] 419: 24,24,24,24,24,31,24,24,24,24,24
[NF2] 420: 0,0,0,0,0,15,0,0,0,0,0
[S13] 421: 8,8,8,8,8,15,8,15,8,8,8
[GU2] 422: 0,0,0,0,0,15,0,15,0,0,0
[0B3] 423: 0,0,0,0,0,248,8,248,8,8,8
[G53] 424: 0,0,0,0,0,240,0,240,0,0,0
[2R3] 425: 0,0,0,0,0,248,24,24,24,24,24
[GU2] 426: 0,0,0,0,0,240,0,0,0,0,0
[ZG3] 427: 0,0,0,0,0,31,24,24,24,24,24

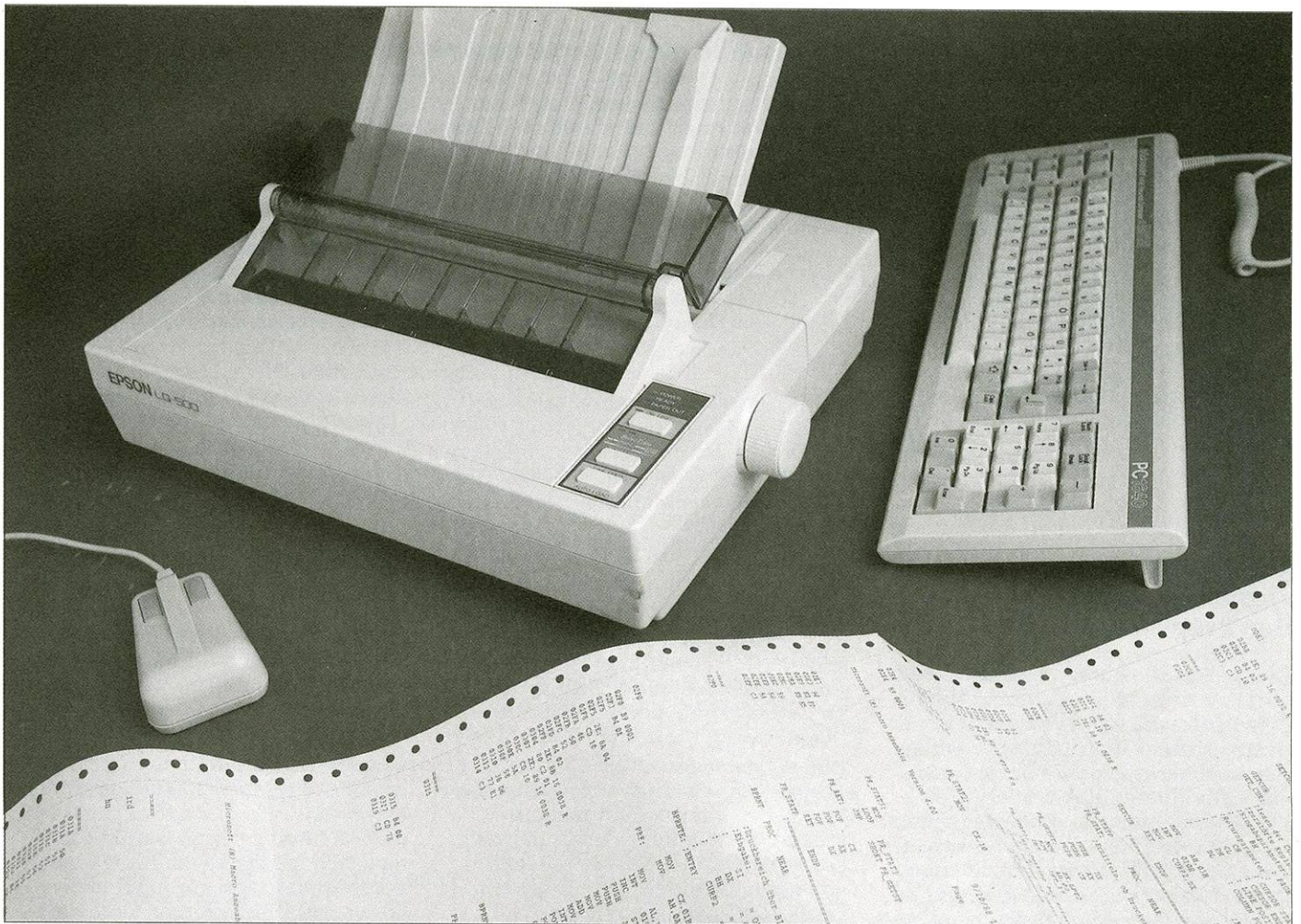
```

```

[TP2] 428: 0,0,0,0,0,31,0,0,0,0,0
[8Y2] 429: 0,0,0,0,0,15,8,15,8,8,8
[GU2] 430: 0,0,0,0,0,15,0,15,0,0,0
[2E3] 431: 8,8,8,8,8,255,8,255,8,8,8
[K63] 432: 0,0,0,0,0,255,0,255,0,0,0
[X84] 433: 24,24,24,24,24,255,24,24,24,24,24
[VU2] 434: 0,0,0,0,0,255,0,0,0,0,0
[GY2] 435: 8,8,8,8,8,248,0,0,0,0,0
[GU2] 436: 0,0,0,0,0,240,0,0,0,0,0
[NT2] 437: 0,0,0,0,0,15,8,8,8,8,8
[NF2] 438: 0,0,0,0,0,15,0,0,0,0,0
[SZ5] 439: 255,255,255,255,255,255,255,255,255,255
[SZ5] 440: 255,255,255,255,255,255,255,255,255,255
[G74] 441: 15,15,15,15,15,15,15,15,15,15
[G74] 442: 15,15,15,15,15,15,15,15,15,15
[PA4] 443: 255,255,255,255,255,255,0,0,0,0,0
[PA4] 444: 255,255,255,255,255,255,0,0,0,0,0
[DB4] 445: 0,0,0,0,0,255,255,255,255,255,255
[DB4] 446: 0,0,0,0,0,255,255,255,255,255,255
[4R5] 447: 240,240,240,240,240,240,240,240,240,240
[WU5] 448: 224,224,224,224,224,224,224,224,224,224
[TZ2] 449: 12,14,2,2,2,10,0,4,8,2,0,0
[PH3] 450: 12,28,16,16,16,0,12,8,4,16,0,0
[625] 451: 127,127,128,148,148,148,148,148,0,104,0,0
[9K3] 452: 127,255,0,0,0,0,0,0,184,64,0,0
[1D3] 453: 0,0,0,62,62,32,32,32,48,0,0,0
[403] 454: 0,0,0,60,60,0,0,0,32,0,0,0
[BC3] 455: 16,16,16,14,14,0,8,0,14,0,0,0
[AD3] 456: 0,0,0,28,28,16,0,16,28,16,0,0
[WA5] 457: 130,198,198,170,170,146,146,130,130,130,0,0
[SW3] 458: 132,132,72,72,48,48,0,0,0,132,0,0
[K14] 459: 28,28,34,34,34,34,32,60,32,32,0,0
[W53] 460: 24,60,0,0,0,0,0,36,24,0,0,0
[S03] 461: 29,31,2,2,2,2,2,2,28,0,0,0
[U22] 462: 29,30,0,0,0,0,0,4,24,0,0,0
[XB3] 463: 0,16,16,16,14,14,0,16,0,16,0,0
[GC3] 464: 16,16,0,0,28,28,16,16,0,0,0
[8T4] 465: 0,130,146,186,130,198,130,170,146,130,0,0
[WH3] 466: 0,0,48,48,72,132,72,0,48,0,0,0
[2H3] 467: 0,0,16,56,16,84,16,56,16,0,0,0
[193] 468: 0,0,48,48,72,72,0,48,0,0,0
[UW3] 469: 18,26,38,32,32,32,32,38,10,18,0,0
[UA3] 470: 16,48,12,0,0,0,0,12,32,16,0,0
[L83] 471: 4,12,66,2,34,2,18,2,8,4,0,0
[EB4] 472: 8,12,16,208,144,176,144,144,4,8,0,0
[GT3] 473: 8,16,16,16,16,8,16,16,16,16,8,0
[743] 474: 16,8,8,8,8,16,8,8,8,16,0
[MT3] 475: 8,16,18,16,20,8,16,48,16,16,8,0
[ME3] 476: 16,8,8,12,8,16,40,8,72,8,16,0
[S04] 477: 0,0,16,56,16,84,16,146,146,130,0,0
[V73] 478: 0,0,48,48,72,0,132,0,0,0,0,0
[615] 479: 126,126,128,128,128,128,128,128,0,126,0,0
[RR3] 480: 124,252,0,0,0,0,0,128,124,0,0,0
[FY3] 481: 18,18,18,18,18,18,18,18,18,18,0,0
[4X2] 482: 8,8,8,8,8,8,8,8,8,8,0,0
[VY3] 483: 18,18,18,18,58,58,18,18,18,18,0,0
[C53] 484: 0,0,0,0,120,120,0,0,0,0,0,0
[183] 485: 2,2,2,2,2,42,42,18,18,2,0,0
[F73] 486: 0,0,0,0,72,72,48,48,0,0,0,0
[N73] 487: 2,2,2,18,18,42,42,2,2,2,0,0
[V73] 488: 0,0,0,0,48,48,72,72,0,0,0,0
[FF3] 489: 0,0,0,0,62,126,0,128,128,0,0,0
[4S3] 490: 0,0,0,0,124,124,128,0,128,0,0,0
[D63] 491: 4,0,2,2,0,252,248,0,0,0,0,0
[W63] 492: 0,4,0,0,4,248,248,0,0,0,0,0
[7V2] 493: 0,0,0,0,36,36,0,0,0,0,0,0
[8X3] 494: 16,16,16,16,84,84,16,16,16,16,0,0
[K83] 495: 4,4,4,32,32,4,4,4,32,32,0,0
[5D3] 496: 32,32,32,8,8,32,32,32,8,8,0,0
[BR3] 497: 0,0,0,64,128,128,128,64,0,0,0,0
[5H3] 498: 0,0,0,128,64,64,64,128,0,0,0,0
[DV2] 499: 0,0,0,0,16,16,0,0,0,0,0,0
[DV2] 500: 0,0,0,0,16,16,0,0,0,0,0,0
[MN2] 501: 0,0,0,0,0,0,0,0,0,0,0,0
[4R2] 502: 0,0,0,0,0,16,0,0,0,0,0,0
[HD3] 503: 0,0,0,8,8,4,4,2,2,254,128,128
[VD3] 504: 16,16,16,16,8,8,4,4,0,252,0,0
[E03] 505: 224,128,128,128,96,0,0,0,0,0,0,0
[063] 506: 224,0,0,0,224,0,0,0,0,0,0,0
[S63] 507: 0,160,128,64,0,0,0,0,0,0,0,0
[ZA3] 508: 160,32,96,160,0,0,0,0,0,0,0,0
[S63] 509: 0,0,0,0,24,24,24,24,0,0,0,0
[K63] 510: 0,0,0,0,16,16,16,16,0,0,0,0
[MN2] 511: 0,0,0,0,0,0,0,0,0,0,0,0
[MN2] 512: 0,0,0,0,0,0,0,0,0,0,0,0

```

Listing 5. Die Zeichensatzdatei »double.tbl«



So arbeitet ein Gerätetreiber

Das MS-DOS-Betriebssystem ist ein offenes Betriebssystem. Damit ist gemeint, daß es sich fast grenzenlos erweitern läßt. Nicht zuletzt ist das ein Grund für den großen Erfolg von MS-DOS auf dem Markt. Anwenderspezifische Erweiterungen oder Anpassungen werden über genau definierte Schnittstellen eingebracht. Eine dieser Schnittstellen ist die für Gerätetreiber. Doch was ist überhaupt ein Gerätetreiber, und wie arbeitet er?

Sicherlich haben Sie sich schon einmal auf Ihrer Systemdiskette oder Festplatte die Datei »config.sys« angesehen oder gar Änderungen daran vorgenommen, beispielsweise den folgenden Eintrag erzeugt:

```
device = ansi.sys
```

Danach wurde der Rechner von Ihnen ganz nach Vorschrift neu gestartet (»gebootet«). Vielleicht kam dabei der Wunsch auf zu erfahren, was es mit einem solchen Programm wie »ansi.sys« auf sich hat, das so ganz anders behandelt wird als irgendein Anwenderprogramm. Vielleicht hätten Sie so etwas auch gerne selbst

programmiert, da Bedarf dafür durchaus vorhanden gewesen wäre. Aus Mangel an Informationen mußten Sie sich jedoch unter Umständen mit einem Brückenprogramm zufriedengeben. Was also verbirgt sich hinter einem solchen Programm, das durch einen Eintrag in der Konfigurationsdatei vom Betriebssystem geladen wird? Wie sieht es aus, und wie programmiert man es?

Ein solches Programm ist ein Gerätetreiber oder im englischen Original: ein »Device Driver«. Er läßt sich nicht wie andere Programme durch Angabe seines Namens ausführen. Der An-

wender hat überhaupt keine Eingriffsmöglichkeit. Vielmehr lädt das Betriebssystem dieses Programm, das nun nicht mehr als Programm schlechthin, sondern vielmehr als Erweiterung des Betriebssystems behandelt wird. Ein Gerätetreiber wird zur Laufzeit auch fast ausschließlich vom Betriebssystem aufgerufen. Eine Ausnahme stellt die sogenannte IOCTL-Schnittstelle dar, über die eine Anwendersoftware direkt, unter Umgehung des MS-DOS, mit dem Treiber kommunizieren kann. Dabei können allerdings keine Daten, sondern nur Steuerinformationen wie zum Beispiel die Abfrage nach dem Gerätestatus oder ähnlichem übermittelt werden.

Im weiteren Verlauf werden Sie nach einigen Vorüberlegungen die Grundlagen und den theoretischen Hintergrund zur Herstellung von Gerätetreibern beschreiben finden. Gleichsam als Einstiegsprojekt erfolgt dann die Programmierung eines Treibers, der einen Drucker emulieren kann. Die Emulation läßt sich an Ihren Drucker anpassen, die Wahl der Druckerports ist dynamisch veränderbar.

Doch zunächst einige Grundlagen: Ein Gerätetreiber wird von MS-DOS

konsequenterweise im unteren Adreßbereich abgelegt, in dem sich das Betriebssystem selbst befindet. Gerätetreiber sind somit als Bindeglieder zwischen der möglichst hardwareunabhängigen Schnittstelle des Betriebssystems und der Hardware selbst zu verstehen.

Mit Einführung von MS-DOS 2.0 wurde die Schnittstelle zum Betriebssystem systemtechnisch fehlerfrei und umfassend dokumentiert und definiert. Dabei wurde der Begriff des »installable Device Drivers« geprägt. Wenn heute im Zusammenhang mit kompatiblen Computern von einem Gerätetreiber gesprochen wird, so ist damit eben dieser »installable Device Driver« gemeint, der gemäß der Schnittstellendefinitionen von Microsoft erstellt ist.

Gerätetreiber sind komplizierter als normale Anwender-Programme

Diese Definition zeigt, daß ein Gerätetreiber regelmäßig hardwareabhängig ist, zum Beispiel von einem Drucker, einer Festplatte oder anderen peripheren Einheiten, zu denen auch ein Plotter oder ein Schrittmotor gehören könnten. Demgegenüber ist seine Schnittstelle zum Betriebssystem hin vollkommen unabhängig von der Peripherie und dem Computertyp, solange MS-DOS das Betriebssystem ist. Diese Tatsache ist gar nicht so selbstverständlich. Andere Betriebssysteme bieten keine solche Schnittstelle, oder es ist an die betreffende Dokumentation nicht heranzukommen, was für einen Anwender bedeutet, daß Erweiterungen nur unter hohem finanziellen Aufwand oder überhaupt nicht zu erhalten sind. Es ist sicherlich bereits deutlich geworden, daß der Code eines Gerätetreibers komplizierter ist als der eines normalen Programms, weil der Gerätetreiber die Konventionen des Be-

Anatomie eines Gerätetreiberaufrufs

Beim Aufruf eines Gerätetreibers reserviert MS-DOS etwa 40 bis 50 Byte für den Stack. Der Gerätetreiber ist dafür verantwortlich, daß der Prozessor-Status durch den Aufruf des Treibers nicht verändert wird. Alle benutzten Register inklusive des Flag-Registers müssen daher gerettet und vor Rückkehr zu MS-DOS restauriert werden. Ein Aufruf spielt sich wie folgt ab:

- MS-DOS schreibt einen passenden Request-Header in einen reservierten Speicherbereich.
- MS-DOS ruft die strategische Routine des Gerätetreibers mit einem »Far Call« auf und übergibt die Adresse des Request-Headers in den Registern »ES:BX«
- Der Gerätetreiber sichert die übergebene Adresse des Request-Headers und führt ein »Far Return« aus. Weitere Aktionen sind dem Gerätetreiber an dieser Stelle untersagt.
- MS-DOS ruft die Interrupt-Routine des Gerätetreibers auf.
- Der Gerätetreiber holt den gesicherten Zeiger auf den Request-Header und liest den Kommando-Code (Offset 2 im Request-Header).
- Falls es sich um einen blockorientierten Treiber handelt, wird der Unit-Code gelesen, um festzustellen, welches physikalische Gerät angesprochen werden soll.
- Der Gerätetreiber holt sich, falls erforderlich, zusätzliche Informationen aus den Feldern des erweiterten Request-Headers.
- Der Gerätetreiber führt die angeforderte Aktion aus.
- Der Gerätetreiber setzt Bit 8 des Status-Wortes im Request-Header, um MS-DOS anzuzeigen, daß die angeforderte Aktion beendet wurde.
- Falls ein Fehler aufgetreten ist, wird zusätzlich das Fehler-Bit (Bit 15) im Status-Wort gesetzt und ein Fehlercode ebenfalls im Statuswort festgehalten.
- Der Gerätetreiber kehrt mit einem »Far Return« zu MS-DOS zurück.

triebssystems zu erfüllen hat. Es drängt sich die Frage auf, warum es nicht auch ein Anwendungsprogramm tun könnte. Es gibt nun trotz des höheren Aufwandes eine Reihe von guten Gründen, die für die kompliziertere Lösung sprechen.

Da wäre zum einen die enge Verbindung zu nennen, die ein Gerätetreiber mit dem Betriebssystem eingeht. Letztendlich wird er zu einem Teil des Betriebssystems, von dem er beim Booten auch ohne Zutun eines Benutzers geladen wird. Der Eintrag in der Datei »config.sys« reicht dafür aus.

Des weiteren wirken alle MS-DOS-Befehle, wie bei MS-DOS dokumentiert, auf den Gerätetreiber, und selbstverständlich gelten auch alle Schnittstellen für jede Art von An-

wendersoftware inklusive der Compiler. So sorgt MS-DOS auch für einen korrekten Aufruf und für die passende Systemumgebung. Darum muß man sich bei der Herstellung eines Gerätetreibers fast gar nicht mehr kümmern. Diesen hohen Grad der Vereinheitlichung und der Sicherheit beim Betrieb erreicht man nur durch eine Erweiterung des Betriebssystems, eben durch einen Gerätetreiber.

Unter MS-DOS werden zwei grundsätzliche Arten von Gerätetreibern unterschieden, nämlich die zeichenorientierten Gerätetreiber (»Character Device Drivers«) und die blockorientierten Gerätetreiber (»Block Device Drivers«). Je nach Anwendung wählt man den einen oder den ande-

Wer — wo — wann?

Bei Zuschriften bitten wir, folgendes zu beachten: Haben Sie Fragen, Anregungen oder Programme, die Sie der DOS-Redaktion zukommen lassen möchten, adressieren Sie wie folgt:

**Redaktionsbüro Everts & Hagedorn · Redaktion »DOS International«
Hans-Stießberger-Str. 3 · 8013 Haar/München · Tel. (089) 46 10 01-0**

Haben Sie Fragen, die zum Beispiel Abonnements, Kleinanzeigen oder Databox-Bestellungen betreffen, wenden Sie sich bitte an den Verlag:

**DMV-Verlag GmbH · DOS International · Fuldaer Straße 6 · 3440 Eschwege
Tel. (05651) 8009-0 · Telex 993 210 dmv d · Telefax (05651) 8009-33**

ren Typ. Für die Steuerung von Druckern, Bildschirmen und Tastaturen wird man sich für einen zeichenorientierten Gerätetreiber entscheiden, da bei dieser Peripherie in der Regel zeichenweise (Zeichen für Zeichen) übertragen wird. Auf diesen Typ soll im weiteren besonders eingegangen werden, denn als praktische Anwendung ist ja ein Gerätetreiber herzustellen, der einen Drucker emulieren kann. Für die Steuerung von Diskettenlaufwerken und Festplatten wählt man dagegen einen blockorientierten Gerätetreiber. Hier werden die Daten in der Regel nämlich blockweise übertragen. Der Unterschied in der Struktur der beiden Typen ist nicht besonders groß. Allerdings müssen einige Besonderheiten sehr wohl beachtet werden.

Ein Gerätetreiber hat einen besonderen Programmaufbau. Diese Struktur ist zwingend vorgeschrieben. Sie beginnt mit einem sogenannten »Device Header« dessen Aufbau und Lage im Programm fest vorgeschrieben ist. Weiterhin ist ein Initialisierungsteil enthalten, der an das Ende des ge-

samten Codes gelegt wird. Das ist zwar nicht zwingend vorgeschrieben, jedoch deswegen sinnvoll, weil diese Codestrecke nur ein einziges Mal, nämlich bei der Installation des Treibers beim Starten von MS-DOS durchlaufen wird. Liegt sie nun am Ende, kann dieser Hauptspeicherbereich anschließend freigegeben werden, was den effektiven Hauptspeicherbedarf zur Laufzeit natürlich verringert. Des weiteren wird ein Befehlsteiler benötigt, der in seiner Form und Lage nicht vorgeschrieben ist.

Die Struktur eines Gerätetreibers ist zwingend vorgeschrieben

Der in Listing 1 vorgestellte Teiler ist systemtechnisch vorteilhaft und »sauber«. Außerdem erlaubt diese Form die Herstellung eines generellen Treiber-Skeletts, in das nur noch die Codestrecken der einzelnen Funktionen einzutragen sind. Listing 1 zeigt die einzelnen Bausteine und deren Anordnung in stilisierter Form. Die Bedeutung der einzelnen

Felder ist in einer Übersicht in Tabelle 1 dargestellt. Die Bedeutung dieser Felder im einzelnen ist nun aber von allerhöchster Wichtigkeit und ein recht »trockenes« Kapitel. – Vor den Erfolg haben die Götter eben den Schweiß gesetzt. Betrachten Sie also im folgenden die Felder in Tabelle 1 und ihren theoretischen Hintergrund.

MS-DOS besitzt von Haus aus bereits einige Gerätetreiber. Dazu gehören unter anderem die Treiber »con«, »aux«, »com1«, »prn«, »lpt1«, und nicht zuletzt »ansi.sys«. MS-DOS verwaltet deren Adressen in einer Kette von Adreßverweisen. Selbstverständlich wird auch jeder selbstgeschriebene Gerätetreiber sich darin einordnen müssen. Im Link- oder Kettungsfeld werden nur dann Adressen angegeben, wenn der Code mehrere Einsprungspunkte, also mehrere Gerätetreiber enthält. Enthält er nur einen, wie in Listing 1, so werden beide Worte des Feldes auf »-1« gesetzt. Sollen mehrere Treiber in einem einzigen Listing angelegt werden, so enthält das erste Wort des Link-Feldes die Offset-Adresse und das zweite natürlich die Segment-Adresse des nachfolgenden Treibers. Das Kettungsfeld des dann letzten Treibers in dieser Kette muß selbstverständlich wieder das obligatorische »-1,-1« enthalten. Mehr ist bezüglich des Kettungsfeldes nicht zu beachten.

Das Attributfeld ist ein Wort lang und wird bitweise besetzt. Jedes Bit hat eine ganz besondere und genau festgelegte Bedeutung. MS-DOS liest die Attribute und reagiert dementsprechend. Sollten sie nicht richtig gesetzt sein, wird man sich über das Verhalten des Betriebssystems sehr wundern. In Tabelle 2 ist die Bedeutung der einzelnen Bits angegeben. Die tiefere Bedeutung der Attribute wird später bei der Behandlung der Funktionen deutlich, die der Gerätetreiber im Auftrag von MS-DOS auszuführen hat.

Die Interrupt-Routine eines Gerätetreibers führt in der Regel die Arbeit aus

Es folgen zwei Adreßfelder. »ms_strat« enthält die Adresse des Einsprungpunktes in die »strategische Routine« (»Strategy Routine«) und »ms_int« die in die Interrupt-Routine des Gerätetreibers. Diese Adressen sind relativ zum aktuellen Segment, so daß eine Segment-Adresse nicht angegeben werden muß. Wenn MS-DOS einem Treiber einen Auftrag erteilt, so setzt es einen »Far Call« auf die strategische Routine ab. Dabei übergibt es dem Treiber in den Registern »ES:BX« die Adresse

```

;===== Device Header =====
;
DEV_HDR:
    DW      -1,-1                ;LINK-Feld
    DW      0000000000000000B    ;Attribute
    DW      MS_STRAT              ;STRATEGY ENTRY
    DW      MS_INT               ;INTERRUPT ENTRY
    DB      "LPT2"               ;DEVICE NAME
;=====
;
;===== Befehlsteiler =====
;
COMM_TBL:
    DW      MS_INIT              ;INIT
    DW      MS_ERR1              ;MEDIA CHECK
    DW      MS_ERR1              ;BUILD BPB
    DW      MS_ERR1              ;IOCTL, NOT POSSIBLE
    DW      MS_ERR1              ;INPUT
    DW      MS_ERR1              ;NON DESTRUCTIVE INPUT
    DW      MS_ERR1              ;INPUT STATUS
    DW      MS_ERR1              ;FLUSH BUFFER
    DW      MS_ERR1              ;OUTPUT
    DW      MS_PRINT             ;OUTPUT MIT VERIFY
    DW      MS_STA               ;OUTPUT STATUS
    DW      MS_ERR2              ;OUTPUT FLUSH
    DW      MS_ERR1              ;IOCTL, OUTPUT
    DW      MS_ERR1              ;DEVICE OPEN
    DW      MS_ERR1              ;DEVICE CLOSE
    DW      MS_ERR1              ;REMOVABLE MEDIA
    DW      MS_PRINT             ;OUTPUT UNTIL BUSY
;=====
;
;===== Initialisierungsteil =====
;
;Hier folgt der Code für die Initialisierung beim Boot von
;MS-DOS und die Freigabe von Hauptspeicherbereichen, die
;zur Laufzeit nicht mehr benötigt werden. Weiterhin führt
;der Driver hier seine spezifischen Initialisierungen durch.
;Dieser Teil muß von Ihnen für jeden Treiber selbst erstellt
;werden und ist deshalb nicht abgedruckt.
;=====

```

Listing 1. Die einzelnen Bestandteile eines Gerätetreibers

zu einem sogenannten »Request Header«. Der liegt irgendwo in einem Hauptspeicherbereich des MS-DOS und stellt so etwas wie ein »Kommunikationspaket« dar, dessen Aufbau wiederum, wie könnte es auch anders sein, wohldefiniert ist. Darüber hinaus hängt sein Aufbau von der aktuellen Funktion ab. Der Treiber hat jetzt die Aufgabe, die Adresse des Request-Headers zu sichern. Es ist ihm verboten, irgendwelche andersgearteten Operationen auszuführen, insbesondere keine Ein- und Ausgabeoperationen. Der Treiber gibt also die Kontrolle sofort wieder an das Betriebssystem zurück.

MS-DOS ruft unmittelbar danach die Interrupt-Routine des Gerätetreibers auf. Dort wird die eigentliche Arbeit des Treibers verrichtet. Nachdem der Treiber die Kontrolle erhalten hat, inspiziert er den Request-Header, dessen Adresse er ja vorher gerettet hatte. Dabei erfährt er, was das Betriebssystem überhaupt von ihm will, also welche Funktion MS-DOS von ihm verlangt. Er führt die geforderten Aktionen aus oder nicht, wenn es nicht möglich ist, und gibt einen Antwortcode zurück.

Der Gerätetreiber muß sich mit MS-DOS verständigen. Einmal benötigt er Angaben darüber, was MS-DOS von ihm möchte, zum anderen muß er eine Rückmeldung erstatten. Wenn diese Rückmeldung unterbleibt, kann das ganze System auf einen Schlag stehenbleiben. Drittens braucht der Treiber für einige Funktionen besondere Angaben, zum Beispiel dann, wenn er für MS-DOS ein Zeichen drucken soll. Dann muß er nämlich wissen, wo er dieses Zeichen findet. Diese Kommunikation erfolgt über den Request-Header. Sein Aufbau ist in Tabelle 3 dargestellt.

Jetzt wird es wieder etwas »trocken« – die Felder des Request-Headers in Tabelle 3 müssen genauer betrachtet werden. MS-DOS kann von einem Gerätetreiber theoretisch maximal die Ausführung von 17 verschiedenen Funktionen verlangen. Hierbei handelt es sich um die für MS-DOS allgemeingültig veröffentlichten Befehls-codes. Im Einzelfall sind es natürlich weitaus weniger, da einige Funktionen nur für den einen Typ zugelassen sind, andere dagegen für den anderen. Die Unterschiede in der Funktionalität bedingen, daß auch die Parameterversorgung unterschiedlich ist. Wenn der Treiber nur den Status eines Gerätes ermitteln soll, so braucht er in der Regel auch nur ein Wort für diesen an MS-DOS zurückzugeben - den Antwortcode eben. Wenn er aber einen Block von der Platte lesen soll, dann benötigt er eine Menge an Informationen mehr. Er muß wissen, von welcher Platte,

Feldname	Offset	Feldtyp	Bedeutung
K-Feld	0	DWord	Kettungsfeld. MS-DOS verkettet alle Gerätetreiber. Ein Wert von (-1,-1) bedeutet, daß es sich um den letzten Gerätetreiber in einer Kette handelt.
Attribute	4	Word	Hier werden die Eigenschaften des Gerätetreibers definiert. MS-DOS interpretiert diese Bits und erteilt dem Gerätetreiber dementsprechend Aufträge. Hier wird zum Beispiel auch definiert, ob es sich um ein zeichen- oder ein blockorientiertes Gerät handelt.
Strategy Entry	6	Word	Das ist die Adresse des Einsprungspunktes der Strategy-Routine im Gerätetreiber. Diese Routine wird von MS-DOS aufgerufen.
Interrupt Entry	8	Word	Das ist die Adresse des Einsprungspunktes der Interrupt-Routine in den Gerätetreiber. Diese Routine wird von MS-DOS aufgerufen.
Device Name	10	8 Byte	Das ist der logische Name des Gerätes.
Comm_tbl	18	Code	Befehlssteiler, er enthält Einsprungspunkte in den Gerätetreiber. Für jede Funktion eines Treibers ist ein Einsprungspunkt vorgesehen. Bei einigen Zeilen steht die Zeichenfolge »err«. Sie besagt, daß es sich hierbei um eine für diesen Treiber nicht erlaubte Funktion handelt.

Tabelle 1. Der Aufbau eines »Device Headers« (vergleiche Listing 1)

Bit	Wert	Bedeutung
15	1	Zeichenorientiertes Gerät
	0	Blockorientiertes Gerät
14	1	Treiber hat IOCTL-Schnittstelle
	0	Treiber hat keine IOCTL-Schnittstelle
13	1	Block-Geräte: Treiber verarbeitet auch andere als IBM-Formate (bei Block-Geräten)
	0	Block-Geräte: Treiber verarbeitet nur IBM-Formate
	1	Zeichen-Geräte: Treiber unterstützt »output until busy«
12	–	Zeichen-Geräte: Treiber unterstützt kein »output until busy«
	–	nicht definiert, sollte auf 0 gesetzt werden.
11	1	Treiber unterstützt »open/close« für auswechselbare Speichermedien (nur ab MS-DOS 3.0, unter MS-DOS 2.x sollte es auf 0 gesetzt werden)
	–	nicht definiert, sollte auf 0 gesetzt werden.
10	–	nicht definiert, sollte auf 0 gesetzt werden.
	–	nicht definiert, sollte auf 0 gesetzt werden.
9	–	nicht definiert, sollte auf 0 gesetzt werden.
	–	nicht definiert, sollte auf 0 gesetzt werden.
8	–	nicht definiert, sollte auf 0 gesetzt werden.
	–	nicht definiert, sollte auf 0 gesetzt werden.
7	–	nicht definiert, sollte auf 0 gesetzt werden.
	–	nicht definiert, sollte auf 0 gesetzt werden.
6	1	Treiber unterstützt MS-DOS 3.2.
	0	Treiber unterstützt nur MS-DOS-Versionen unterhalb von 3.2
5	–	nicht definiert, sollte auf 0 gesetzt werden.
	–	nicht definiert, sollte auf 0 gesetzt werden.
4	–	nicht definiert, sollte auf 0 gesetzt werden.
	–	nicht definiert, sollte auf 0 gesetzt werden.
3	1	Treiber ist aktuelles »Clock-Device« (Uhr)
	–	Treiber ist »Nul Device« (Datensenke)
2	1	Treiber ist »Standard Output Device« (Konsole)
	–	Treiber ist »Standard Input Device« (Konsole)
1	1	Treiber ist »Standard Output Device« (Konsole)
	–	Treiber ist »Standard Input Device« (Konsole)
0	1	Treiber ist »Standard Input Device« (Konsole)
	–	Treiber ist »Standard Output Device« (Konsole)

Tabelle 2. Das Attributfeld für Gerätetreiber

Feldname	Offset	Feldtyp	Bedeutung
length	0	Byte	Gesamtlänge des Request-Headers
unit	1	Byte	Unit-Nummer für diesen Aufruf
command	2	Byte	Befehlscode für den Treiber
status	3	Word	Antwortcode, Status
reserved	5	8 Byte	(reserviert)
media	13	Byte	»Media Descriptor Byte«
adress	14	DWord	Pufferadresse für Datentransfer
count	18	Word	Zeichen-/Sektorzähler
sector	20	Word	Startsektor

Tabelle 3. Aufbau des erweiterten Request-Headers

von welcher Stelle ab er lesen soll und wo er die gelesenen Daten abliefern soll. Alle diese notwendigen Parameter werden im Request-Header festgehalten. Sie werden von dem Treiber einerseits und von MS-DOS andererseits nach festgelegten Regeln interpretiert. In Tabelle 3 ist der erweiterte Request-Header dargestellt, das heißt, es sind alle Felder enthalten, die vorkommen können. Der eigentliche, immer gleiche Kern des Request-Headers umfaßt die ersten 13 Byte. Da die Parameterversorgung funktionsabhängig ist, werden für die jeweils aktuelle Funktion nur die benötigten Felder beschrieben.

Im Feld »Laenge« wird die Länge des Request-Headers in Zeichen angegeben. Das ist dann von Bedeutung, wenn eine Funktion den erweiterten Request-Header benötigt.

In »unit« wird die Gerätenummer übergeben. MS-DOS numeriert die Geräte nämlich intern. So hat die erste Diskettenstation die Nummer 0 und so weiter. Es ist nun denkbar, daß zum Beispiel ein Gerätetreiber zwei Diskettenlaufwerke betreiben kann. In diesem Fall würde er im Feld »unit« erkennen können, für welches Laufwerk eine Bearbeitung verlangt ist. »unit« ist nur für Block-Gerätetreiber bedeutsam.

»command« als Sprachorgan für die Verständigung zwischen MS-DOS und dem Treiber

Im Feld »command« gibt MS-DOS den Befehlscode bekannt. Einfacher ausgedrückt: MS-DOS sagt, was es vom Treiber will. Je nachdem, um welchen Typ es sich handelt, gibt es eine wohldefinierte Menge an Funktionen, die ein Treiber ausführen können muß. Die Funktionscodes unter MS-DOS 3.x können die Werte zwischen 0 und 16 annehmen. Je nach dem Typ des Treibers und den Attributen sind einige von ihnen nicht erlaubt. Im Befehlssteiler in Listing 1 finden Sie genau 17 Einträge; sie sind entsprechend den Funktionscodes angeordnet. Ihre Adressen verweisen auf die Einsprungspunkte der entsprechenden Codestrecken im Treiber. Dort, wo der Zusatz »err1« oder »err2« angebracht ist, liegen ungültige Funktionscodes vor, und es wird in eine Fehleroutine des Treibers verzweigt. Normalerweise kann ein Gerätetreiber aber nicht auf einen dieser Ausgänge kommen. Wäre dies nämlich der Fall, hätte MS-DOS einen falschen Befehlscode angegeben. Die durchgängige Deklaration aller möglichen Codes hat einen Vorteil: Programmtechnisch gesehen, läßt sich der Einsprungspunkt in den Be-

Bit	Bedeutung (wenn Bit gesetzt)
15	Treiber meldet eine Fehlerbedingung
14	(reserviert)
13	(reserviert)
12	(reserviert)
11	(reserviert)
10	(reserviert)
9	Treiber ist beschäftigt und kann keinen weiteren Auftrag übernehmen
8	Treiber hat den gegenwärtigen Auftrag erfüllt.
0 bis 7	Fehlercodes (siehe Tabelle 5)

Tabelle 4. Statuscode vom Gerätetreiber

Code	Bedeutung
0	Schreibschutz-Verletzung
1	Unit ist nicht bekannt
2	Laufwerk ist nicht bereit
3	Unbekannter Befehlscode
4	Datenfehler, zum Beispiel CRC-Fehler bei einer Diskette
5	Ungültige Struktur des Laufwerk-Request-Headers
6	»seek error« bei Massenspeicheroperation
7	Unbekanntes Speichermedium
8	Sektor kann nicht lokalisiert werden
9	Papierende beim Drucker
10	Fehler beim Schreiben
11	Fehler beim Lesen
12	Genereller Fehler, nicht näher definierbar
13	(Reserviert)
14	(Reserviert)
15	Unerlaubter Diskettenwechsel (nur MS-DOS 3.x)

Tabelle 5. Fehlercodes vom Gerätetreiber

fehlsteiler mit Hilfe des Befehlscodes leichter berechnen. Außerdem kann man mit den vollständigen Codes ein Muster für jede Art von Gerätetreiber erstellen.

Im Status-Feld meldet der Gerätetreiber zur Abwechslung etwas an das Betriebssystem. Es handelt sich dabei um einen Antwortcode, der allerdings auch wieder wohldefiniert ist. In den Bits 0 bis 7 wird ein etwaiger Fehlercode angegeben, in 8 bis 15 der Statuscode. Tabelle 4 zeigt die Statuscodes, Tabelle 5 die Fehlercodes. Zu beachten ist, daß im Falle eines Fehlers der Fehlercode einzutragen ist. Zusätzlich ist aber das Fehler-Bit (Bit 15) zu setzen. Wenn das Fehler-Bit nicht gesetzt wird, ist für MS-DOS die Welt in Ordnung, selbst wenn der Fehlercode einen Fehler ausweist.

Das Feld »reserved« ist grundsätzlich reserviert. »Media« wird nur bei Block-Gerätetreibern verwendet. Die Einzelheiten werden später bei der Behandlung der Funktion 1 (»Media Check«) erklärt. »Adress«, »Count« und »Sector« sind ebenfalls vom Befehlscode abhängig und werden zu einem späteren Zeitpunkt behandelt.

Wenden Sie sich nun wieder Listing 1 zu. Das letzte Feld des Device-Headers ist das Feld »Device_Name«. Es ist acht Byte lang und enthält den logischen Namen des Geräts. Damit ist nicht der Name des Gerätetreibers gemeint! Vielmehr wird hier die Bezeichnung vergeben, unter der MS-DOS das Gerät kennen soll. In Li-

sting 1 wurde der Name »lpt2« zugewiesen. Ebenso gut hätte das Gerät den Namen »lpt1« erhalten können; dann hätte MS-DOS es als Standarddrucker übernommen. Sie werden aber noch sehen, daß auch »lpt2« über die erste Druckerschnittstelle druckt und dabei die von Ihnen gewünschte Emulation durchführt. Bei der Vergabe des logischen Namens spielt diese Schnittstelle nämlich zunächst keine Rolle. Sie betrifft die Hardware. Der Name dagegen bedeutet für MS-DOS die Existenz eines solchen Geräts. Am Beispiel von Listing 1 würde der Befehl

```
copy config.sys lpt2
```

bewirken, daß die Datei »config.sys« durch den selbstgeschriebenen Gerätetreiber »lpt2« gedruckt wird. Auf welchem Druckerport der Treiber arbeitet und welche Emulation er durchführt, ist MS-DOS vollkommen gleichgültig – es weiß davon nichts. Auf dieser logisch orientierten Seite ist MS-DOS völlig hardwareunabhängig.

Sie haben nun bereits einen Großteil der Theorie der Gerätetreiber kennengelernt. Vieles allerdings ist noch offen, was in der Hauptsache aber mit dem jeweiligen Funktionscode zusammenhängt. Diese Fragen werden in Verbindung mit den Abläufen der einzelnen Funktionen in der nächsten Folge geklärt.

(M. Schmidt/ev)

Kommandos für Gerätetreiber

Ein Gerätetreiber wird fast ausschließlich von MS-DOS aufgerufen. Die Menge der Befehlscodes und die Art und Weise, wie er dem Betriebssystem gegenüber zu reagieren hat, sind von Microsoft definiert und dokumentiert. Dieser Artikel behandelt den Ablauf des Aufrufs des Treibers, die Entwicklung eines generellen Codeskeletts und die Vorschriften bei der Abarbeitung der einzelnen Befehlscodes.

Wenn MS-DOS gestartet wird, liest es zuerst die Datei »config.sys«. Jeder dort eingetragene Gerätetreiber wird nach und nach geladen. Seine Adressen und Namen »merkt« sich das Betriebssystem in internen Tabellen und Adreßketten. Bei diesem Ladevorgang wird ein Gerätetreiber das erste Mal aufgerufen, um eine Initialisierung zu ermöglichen. Jeder Aufruf läuft nach dem gleichen Schema in zwei Stufen ab.

MS-DOS setzt als erstes einen »Far Call« auf die »strategische Routine« des Gerätetreibers. Der Treiber muß die Inhalte der Register »ES:BX« speichern, denn sie enthalten die Adresse zum Request-Header, einem Bereich, der von dem Betriebssystem für jeden Aufruf neu festgelegt wird. Der Treiber gibt die Kontrolle unverzüglich wieder an MS-DOS zurück und erwartet einen nachfolgenden »Far Call« auf seine Interrupt-Routine. Dort rettet er die Register und lädt die vorher gerettete Adresse des Request-Headers. Nun wendet er sich dem Request-Header zu. Zunächst interessiert der Funktionscode. Der Treiber liest diesen, verzweigt in den entsprechenden Codeteil, führt die Ak-

tionen aus, stellt die Registerinhalte wieder her, schreibt einen Antwortcode in den Request-Header und gibt schließlich die Kontrolle wieder an MS-DOS zurück. Dieser Ablauf ist hinsichtlich seiner Systematik für jeden Befehlscode gleich.

Das allgemeine Codeskelett eines Gerätetreibers kann immer wieder verwendet werden

Da jeder Aufruf des Gerätetreibers in der gleichen Art erfolgt, liegt es nahe, ein generelles Skelett zu entwerfen, in das nur noch anwendungsspezifische Codestrecken einzutragen sind. Bereits in der letzten Folge haben Sie einige Teile eines Treibers gesehen, den »Device-Header«, den Befehlsteiler und den Initialisierungsteil. Wenn Sie diese Teile zusammenfügen und mit etwas »Drumherum« programmieren, so entsteht daraus ein fertiges Skelett, das für jeden Gerätetreiber verwendet werden kann. In Listing 1 finden Sie es nochmals abgedruckt. Selbstverständlich ist das noch kein vollständi-

ger Gerätetreiber; es fehlen schließlich die anwendungsspezifischen Teile. Der Code beginnt mit dem Device-Header bei der Offset-Adresse 0. Der Befehlsteiler enthält Eintragungen für jeden vorkommenden Befehlscode und verweist auf die entsprechende Marke. An Variablen ist augenblicklich nur »r_pak« definiert, worin die Adresse des Request-Headers des jeweiligen Aufrufs gespeichert wird. Die Position der einzelnen Felder innerhalb des Request-Headers ist als Variable mit fester Wertzuweisung angegeben. Die Strategy-Routine in Listing 1 ist vollständig. Über die Return-Routine kehrt jede Funktion zu MS-DOS zurück. Beachten Sie dabei, daß die Return-Routine in einer »Far Proc« liegt und somit den verlangten »Far Return« absetzt.

Die Interrupt-Routine erhält die volle Kontrolle für die Befehlsausführung

Über die Interrupt-Routine erhält der Gerätetreiber von MS-DOS für die Ausführung der einzelnen Befehlscodes die Kontrolle. In der Interrupt-Routine verzweigt er auch über den Befehlsteiler zu den entsprechenden Marken. Die Verzweigung ist bereits vollständig codiert. Danach folgt für jeden Funktionscode eine Marke, unter der nun der anwendungsbezogene Code einzubringen ist. Am Ende liegt der Initialisierungscode unter der Marke »ms_init«. Es gilt jetzt nur noch, in diesem Skelett unter der betreffenden Marke ein »bißchen« Code zu entwerfen. Dazu müssen Sie sich die einzelnen Funktionscodes und den Request-Header genauer ansehen. Oben wurde bereits erwähnt, daß der Gerätetreiber den Request-Header inspiziert, nachdem er den Funktionscode herausgelesen hat. Wozu ist das noch notwendig, da die von MS-DOS verlangte Funktion schließlich schon bekannt ist? Dazu muß gesagt werden, daß das Betriebssystem noch einige andere Parameter übergibt, die von der auszuführenden Funktion abhängen. Diese Parameter werden alle im erweiterten Request-Header übergeben. Es folgt eine ausführliche Besprechung der einzelnen Kommandos, die ein Gerätetreiber ausführen muß. Die Parameterübergabe für die einzelnen Funktionen ist in Tabelle 2 beschrieben.

Funktionscode 0: Gerätetreiber initialisieren. Wenn MS-DOS gestartet wird, liest es unter anderem die Datei »config.sys« und lädt nach und nach jeden dort eingetragenen Gerätetreiber. Wenn dieser geladen ist, bekommt er den Auftrag erteilt, seine

Offset	Typ	Bedeutung
0	Word	Anzahl Bytes pro Sektor
2	Byte	Anzahl Sektoren pro Cluster
3	Word	Anzahl reservierter Sektoren
5	Byte	Anzahl FATs auf dem Datenträger
6	Word	Maximale Anzahl von Einträgen im »Root-Directory«
8	Word	Gesamtzahl vorhandener Sektoren
10	Byte	Media-Descriptor-Byte
11	Word	Anzahl Sektoren, die eine einzelne FAT belegt

Tabelle 1. Der Aufbau des BIOS-Parameter-Blocks (BPB)

Funktionscode 0: Init			
Funktion: Initialisierung des Gerätetreibers			
Eingabeparameter:			
RH + 2	BYTE	Funktionscode 0	
RH + 18	DWORD	Zeiger auf Gleichheitszeichen in der Datei »config.sys«	
RH + 22	BYTE	Gerätenummer (Block-Geräte)	
Rückgabeparameter:			
RH + 3	WORD	Status- und Fehlercode	
RH + 13	BYTE	Anzahl unterstützter Geräte (Block-Geräte)	
RH + 14	DWORD	Adresse, an der am Ende des Treibers der freie Hauptspeicher beginnt	
RH + 18	DWORD	Zeiger auf BPB (Block-Geräte)	
Funktionscode 1: Media Check			
Funktion: Speichermedium prüfen			
Eingabeparameter:			
RH + 1	BYTE	Gerätenummer	
RH + 2	BYTE	Funktionscode 1	
RH + 13	BYTE	Media-Descriptor-Byte	
Rückgabeparameter:			
RH + 3	WORD	Status- und Fehlercode	
RH + 14	BYTE	Media Change Code (1: Medium nicht gewechselt, -1: Medium gewechselt, 0: unbekannt)	
Funktionscode 2: Build BIOS Parameter Block			
Funktion: BIOS-Parameter-Block erzeugen			
Eingabeparameter:			
RH + 1	BYTE	Gerätenummer	
RH + 2	BYTE	Funktionscode 2	
RH + 13	BYTE	Media-Descriptor-Byte	
RH + 14	DWORD	Pufferadresse (Transfer Area)	
Rückgabeparameter:			
RH + 3	WORD	Status- und Fehlercode	
RH + 18	DWORD	Adresse des neuen BPB	
Funktionscode 3: IOCTL Read			
Funktion: Kontrollinformationen lesen			
Eingabeparameter:			
RH + 1	BYTE	Unit Code (Block-Geräte)	
RH + 2	BYTE	Funktionscode: 3	
RH + 13	BYTE	Media-Descriptor-Byte	
RH + 14	DWORD	Pufferadresse (Transfer Area)	
RH + 18	WORD	Anzahl Zeichen oder Sektoren	
RH + 20	WORD	Startsektor (Block-Geräte)	
Rückgabeparameter:			
RH + 3	WORD	Status- und Fehlercode	
RH + 18	DWORD	Anzahl gelesener Zeichen oder Sektoren	
Funktionscode 4: Read			
Funktion: Daten lesen			
Eingabeparameter:			
RH + 1	BYTE	Gerätenummer (Block-Geräte)	
RH + 2	BYTE	Funktionscode: 4	
RH + 13	BYTE	Media-Descriptor-Byte	
RH + 14	DWORD	Pufferadresse (Transfer Area)	
RH + 18	WORD	Anzahl Zeichen oder Sektoren	
RH + 20	WORD	Startsektor (Block-Geräte)	
Rückgabeparameter:			
RH + 3	WORD	Status- und Fehlercode	
RH + 18	WORD	Anzahl gelesener Zeichen oder Sektoren	
Funktionscode 5: Non-destructive Read			
Funktion: Vorausschauendes Lesen			
Eingabeparameter:			
RH + 2	BYTE	Funktionscode: 4	
Rückgabeparameter:			
RH + 3	WORD	Status- und Fehlercode	
RH + 13	BYTE	Gelesenes Zeichen	
Funktionscode 6: Input-Status			
Funktion: Eingabe-Status melden			
Eingabeparameter:			
RH + 2	BYTE	Funktionscode: 6	
Rückgabeparameter:			
RH + 3	WORD	Status- und Fehlercode	
Funktionscode 7: Flush Input-Buffers			
Funktion: Eingabepuffer löschen			
Eingabeparameter:			
RH + 2	BYTE	Funktionscode: 7	
Rückgabeparameter:			
RH + 3	WORD	Status- und Fehlercode	
Funktionscode 8: Write			
Funktion: Daten schreiben			
Eingabeparameter:			
RH + 1	BYTE	Gerätenummer (Block-Geräte)	
RH + 2	BYTE	Funktionscode: 8	
RH + 13	BYTE	Media-Descriptor-Byte	
RH + 14	DWORD	Pufferadresse (Transfer Area)	
RH + 18	WORD	Anzahl Zeichen oder Sektoren	
RH + 20	WORD	Startsektor (Block-Geräte)	
Rückgabeparameter:			
RH + 3	WORD	Status- und Fehlercode	
RH + 18	WORD	Anzahl gelesener Zeichen oder Sektoren	
Funktionscode 9: Write with Verify			
Funktion: Daten schreiben und prüfen			
Eingabeparameter:			
RH + 1	BYTE	Gerätenummer (Block-Geräte)	
RH + 2	BYTE	Funktionscode: 9	
RH + 13	BYTE	Media-Descriptor-Byte	
RH + 14	DWORD	Pufferadresse (Transfer Area)	
RH + 18	WORD	Anzahl Zeichen oder Sektoren	
RH + 20	WORD	Startsektor (Block-Geräte)	
Rückgabeparameter:			
RH + 3	WORD	Status- und Fehlercode	
RH + 18	WORD	Anzahl gelesener Zeichen oder Sektoren	
Funktionscode 10: Output-Status			
Funktion: Ausgabestatus melden			
Eingabeparameter:			
RH + 2	BYTE	Funktionscode: 10	
Rückgabeparameter:			
RH + 3	WORD	Status- und Fehlercode	
Funktionscode 11: Flush Output-Buffers			
Funktion: Ausgabepuffer löschen			
Eingabeparameter:			
RH + 2	BYTE	Funktionscode: 11	
Rückgabeparameter:			
RH + 3	WORD	Status- und Fehlercode	
Funktionscode 12: IOCTL Write			
Funktion: Kontrollinformationen senden			
Eingabeparameter:			
RH + 1	BYTE	Gerätenummer (Block-Geräte)	
RH + 2	BYTE	Funktionscode 12	
RH + 13	BYTE	Media-Descriptor-Byte	
RH + 14	DWORD	Pufferadresse (Transfer Area)	
RH + 18	WORD	Anzahl Zeichen oder Sektoren	
RH + 20	WORD	Startsektor (Block-Geräte)	
Rückgabeparameter:			
RH + 3	WORD	Status- und Fehlercode	
RH + 18	WORD	Anzahl gelesener Zeichen oder Sektoren	
Funktionscode 13: Open			
Funktion: Geräte-Datei öffnen			
Eingabeparameter:			
RH + 1	BYTE	Gerätenummer (Block-Geräte)	
RH + 2	BYTE	Funktionscode: 13	
Rückgabeparameter:			
RH + 3	WORD	Status- und Fehlercode	
Funktionscode 14: Close			
Funktion: Geräte-Datei schließen			
Eingabeparameter:			
RH + 1	BYTE	Gerätenummer (Block-Geräte)	
RH + 2	BYTE	Funktionscode: 14	
Rückgabeparameter:			
RH + 3	WORD	Status- und Fehlercode	
Funktionscode 15: Removable Media			
Funktion: Prüfung, ob Speichermedium auswechselbar			
Eingabeparameter:			
RH + 1	BYTE	Gerätenummer (Block-Geräte)	
RH + 2	BYTE	Funktionscode: 15	
Rückgabeparameter:			
RH + 3	WORD	Status- und Fehlercode	
Funktionscode 16: Output Until Busy			
Funktion: Daten fortgesetzt schreiben			
Eingabeparameter:			
RH + 2	BYTE	Funktionscode 16	
RH + 14	DWORD	Pufferadresse (Transfer Area)	
RH + 18	WORD	Anzahl Zeichen	
Rückgabeparameter:			
RH + 3	WORD	Status- und Fehlercode	
RH + 18	WORD	Anzahl übertragener Zeichen	

Tabelle 2. Die Funktionscodes für Gerätetreiber (RH steht für »Basisadresse Request-Header«)

Initialisierungen durchzuführen. Dazu kann gehören, daß die Hardware überprüft wird, daß Interruptvektoren gesetzt oder Variablen definiert werden. Der Treiber muß sich und die von ihm kontrollierte Hardware in einen definierten Ausgangszustand versetzen. Werden ihm in der Aufrufzeile in der Datei »config.sys« Parameter übergeben, so muß er diese Zeile lesen.

Zum Beispiel könnte folgender Eintrag

```
device = msprn.sys /0,1
```

in »config.sys« enthalten sein. Dann müßte der Gerätetreiber die Parameter »0« und »1« interpretieren. Das ist aber gar nicht schlimm, denn MS-DOS hat ja diese Zeile bereits gelesen und gibt dem Treiber die Adresse des Gleichheitszeichens nach dem Eintrag »device« bekannt. Diese Adresse ist im Request-Header enthalten. Hat nun der Gerätetreiber seine Initialisierungen erledigt, muß er einige Angaben an das Betriebssystem zurückmelden. Diese Antworten trägt er in bestimmte Stellen des Request-Headers ein.

Jeder Gerätetreiber gibt die Adresse zurück, an der der freie Hauptspeicherbereich beginnt. Diese Adresse nennt man die Break-Adresse. Der Vorgang der Freigabe wird in Listing 1 deutlich. Die Marke »schluss« liegt exakt auf der ersten Adresse des Initialisierungscode. Mit ihm beginnt also der freie Hauptspeicherbereich, da die Initialisierungsroutine nur ein einziges Mal benötigt wird, nämlich beim Neustart des Betriebssystems.

Ohne Request-Header ist ein Block-Gerätetreiber sprachlos

Ein Block-Gerätetreiber muß zusätzliche Eintragungen im Request-Header vornehmen. Er gibt die Anzahl der von ihm unterstützten Geräte an. Wenn zum Beispiel das aktuell letzte Gerät die Bezeichnung »C:« hat, in der Regel also die Festplatte, und der Treiber zwei Geräte unterstützt, dann vergibt MS-DOS die Bezeichnungen »D« und »E«. Der Treiber muß für jedes von ihm unterstützte Gerät einen BIOS-Parameter-Block (BPB) anlegen. Das ist zunächst nichts anderes als ein Bereich von der Länge eines Sektors. Tabelle 1 zeigt den Aufbau des BPB, dessen Felder im einzelnen bei der Funktion »Build BPB« erklärt werden.

Beide Gerätetreiber-Typen können eine Bildschirmmeldung ausdrucken. Sollen weitere MS-DOS-Funktionen verwendet werden, so ist zu beachten, daß nur die Funktionen 01hex bis 0Chex und 30hex angewendet werden dürfen.

Funktionscode 1: Speichermedium prüfen. Die Funktion 1 ist nur bei Block-Gerätetreibern anwendbar. MS-DOS fragt nach dem Status eines Gerätes, wenn es bei Ein- oder Ausgaboperationen damit rechnen muß, daß eventuell der Datenträger gewechselt werden konnte. In einem solchen Fall nämlich müßte es die »File Allocation Table« (FAT) neu einlesen.

Die Problematik ist damit aber längst nicht gelöst. Wenn in den Puffern noch nicht auf die Platte geschriebene Datensätze stehen und MS-DOS die Meldung bekommt, der Datenträger sei gewechselt worden, dann gehen die Änderungen verloren. Erhält MS-DOS vom Treiber dagegen die Auskunft, es sei nicht sicher, daß der Datenträger gewechselt wurde, dann geht MS-DOS davon aus, daß er nicht gewechselt wurde und schreibt die im Puffer befindlichen Datensätze zurück. Das ist natürlich höchst problematisch. Stellen Sie sich vor, die Diskette wäre tatsächlich gewechselt worden – das Ergebnis ist klar, die FAT wäre zerstört.

Ein Block-Gerätetreiber kann so unterschiedliche Speichermedien wie Festplatte und Bandlaufwerk bedienen

Bei einer Festplatte ist ein Wechsel natürlich nie möglich, und der Treiber gibt als »Media Change Code« immer eine »1« zurück. Denkbar dagegen ist ein Wechsel bei Band-, Disketten- oder Wechselplattenstationen. Bei Bandstationen müßte auch ein Rückspulen beachtet werden! Bei Diskettenstationen ist die Problematik offenkundig. Bei Wechselplattensystemen (zum Beispiel »Bernoulli-Boxen«) mit ihrer relativ hohen Kapazität wird es unangenehm, wenn deren FAT zerstört wird. Hier müßte der Algorithmus des Treibers doch etwas mehr »Intelligenz« besitzen als bei einer Diskettenstation. Wenn der Treiber nämlich nicht sicher ist, ob der Datenträger gewechselt wurde, so gibt er eine »0« als »Media Change Code« zurück. MS-DOS schreibt dann eine im Puffer befindliche FAT zurück. Unter Umständen geschieht dies eben auf eine tatsächlich gewechselte Diskette.

Geht der Treiber allerdings davon aus, daß der Datenträger gewechselt wurde, so gibt er eine »-1« zurück. MS-DOS erklärt dann alle im Puffer befindlichen Daten für ungültig. Somit sind diese Daten verloren. Der Verlust weiterer Daten ist nicht zu »beklagen«. Weiterhin führt es dann die Build-BPB-Funktion durch und liest die neue FAT ein.

Funktionscode 2: BIOS-Parameter-Block erzeugen. Diese Funktion wird nur von Block-Gerätetreibern ausgeführt. MS-DOS fordert diese Funktion an, um die Adresse eines gültigen BIOS-Parameter-Blocks (BPB) zu bekommen. Das geschieht in den Fällen, in denen die Überprüfung des Speichermediums mittels Funktion 1 ergeben hat, ein Datenträger sei gewechselt worden oder der Wechsel sei nicht auszuschließen. Der Treiber ermittelt nun einen gültigen BPB und übergibt dessen Adresse an MS-DOS. Dann setzt er das Erledigt-Bit im Status-Wort und gibt die Kontrolle an das Betriebssystem zurück.

Funktionscode 3: Kontrollinformationen lesen. Funktion 3 erlaubt die direkte Kommunikation zwischen einem Anwendungsprogramm und dem Treiber. Es muß das IOCTL-Bit in den Attributen im Device Header gesetzt sein. Diese Funktion ist erst ab MS-DOS 3.0 möglich. Die Kommunikation umfaßt lediglich Kontrollinformationen. Daten werden über eine der Read-Funktionen übertragen.

Funktionscode 4: Daten lesen. Bei Funktion 4 liest der Gerätetreiber Daten von einem Gerät ein und stellt sie dem Betriebssystem in einem Puffer zur Verfügung. Wird ein Fehler erkannt, so wird das Error-Bit gesetzt, ein entsprechender Fehlercode eingetragen und zusätzlich die Anzahl gelesener Bytes oder Sektoren angegeben. Die Rückgabe eines Fehlers allein genügt nicht. Wird die Funktion erfolgreich durchgeführt, so wird das Erledigt-Bit gesetzt, die Anzahl gelesener Bytes oder Sektoren eingetragen, die gelesenen Daten in den Puffer transferiert und die Kontrolle zurückgegeben.

Funktionscode 5: Vorausschauendes Lesen. Die Funktion 5 ist nur für zeichenorientierte Gerätetreiber bedeutsam. Wird ein Block-Gerätetreiber mit dieser Funktion aufgerufen, so gibt er lediglich das Erledigt-Bit zurück. Mit dieser Funktion führt MS-DOS ein »look ahead« durch; es liest also das nächste anstehende Zeichen quasi probeweise ein. Dies kann aber nur dann geschehen, wenn der Gerätetreiber einen Puffer hat, wie zum Beispiel die Tastatur.

In einem solchen Fall kann der Treiber bei der Eingabestatus-Funktion (Funktionscode 6) melden, er habe Zeichen im Puffer, die zur Übertragung bereitstehen. MS-DOS kann nun durch den Funktionsaufruf 5 das nächste Zeichen anfordern, das aber im Puffer erhalten bleibt. Beim nächsten regulären Lesen mittels der Stan-

dardfunktion 4 wird also dieses Zeichen ebenfalls gelesen. Der Treiber gibt dieses Zeichen zurück und setzt das Erledigt-Bit.

Funktionscode 6: Eingabestatus melden. Die Funktion 6 ist nur für zeichenorientierte Gerätetreiber bedeutsam. Wird ein Block-Gerätetreiber mit dieser Funktion aufgerufen, so gibt er lediglich das Erledigt-Bit zurück.

Der Treiber gibt einen Status darüber zurück, ob er auf Eingabe seines Gerätes wartet, oder ob er in seinem Look-Ahead-Puffer die Daten bereits bereithält. Diese Entscheidungsmöglichkeit hat er aber nur dann, wenn er einen solchen Puffer besitzt. Ist dies nicht der Fall, so meldet er im Beschäftigt-Bit stets eine »0« zurück. Wenn er einen Look-Ahead-Puffer besitzt, dann meldet er im Beschäftigt-Bit eine »1« zurück, wenn er dabei ist, einen Zugriff auf sein Gerät zu machen. Die Rückmeldung der »1« veranlaßt MS-DOS, auf den Treiber zu warten.

Funktionscode 7: Eingabepuffer löschen. Die Funktion 7 ist nur für zeichenorientierte Gerätetreiber bedeutsam. Wird ein Block-Gerätetreiber mit dieser Funktion aufgerufen, so gibt er lediglich das Erledigt-Bit zurück. Der Funktionsaufruf weist den Gerätetreiber an, alle in seinem Puffer befindlichen Daten als ungültig zu betrachten und den Puffer neu zu initialisieren.

Funktionscode 8: Daten schreiben. Der Treiber sendet die Daten im Übertragungspuffer an sein Gerät. Die Anzahl der Zeichen bei einem Zeichen-Treiber oder die Anzahl der Sektoren bei einem Block-Treiber findet er unter Offset 18 (relativ zum Request-Header). Block-Treiber finden den Startsektor der Platte unter Offset 20. Es muß das Erledigt-Bit gesetzt werden. Zusätzlich muß die Anzahl der erfolgreich geschriebenen Zeichen oder Sektoren angegeben werden.

Funktionscode 9: Daten schreiben und prüfen. Die Funktion 9 entspricht der Funktion 8 mit dem Unterschied, daß der Gerätetreiber über-

prüft, ob nach dem Schreiben die Daten auch wieder gelesen werden können (»Verify«).

Funktionscode 10: Ausgabestatus melden. Die Funktion 10 ist nur für zeichenorientierte Gerätetreiber bedeutsam. Wird ein Block-Gerätetreiber mit dieser Funktion aufgerufen, so gibt er lediglich das Erledigt-Bit zurück. Funktion 10 arbeitet analog zur Funktion 6. Wenn das Beschäftigt-Bit auf »1« gesetzt ist, dann wartet der Treiber darauf, daß sein Gerät Daten schreibt. Der Treiber kann das Beschäftigt-Bit auf »0« setzen, wenn er garantieren kann, daß er bereit ist, Daten sofort anzunehmen, das heißt, wenn sein Gerät nicht gerade schreibt.

Funktionscode 11: Ausgabepuffer löschen. Die Funktion 11 ist nur für zeichenorientierte Gerätetreiber bedeutsam. Wird ein Block-Gerätetreiber mit dieser Funktion aufgerufen, so gibt er lediglich das Erledigt-Bit zurück. Funktion 11 arbeitet analog zu Funktion 7.

Funktionscode 12: Kontrollinformationen schreiben. Funktion 12 erlaubt die Übermittlung von Kontrollinformationen zwischen einem Gerätetreiber und einer Anwendungssoftware. Diese Funktion kann nur dann aufgerufen werden, wenn das IOCTL-Bit im Attributfeld des Device-Headers eingeschaltet ist.

Funktionscode 13: Geräte-Datei öffnen. Die Open-Funktion ist erst ab MS-DOS 3.0 anwendbar, und nur dann, wenn Bit 11 im Attributfeld des Device-Headers eingeschaltet ist. Ein zeichenorientierter Gerätetreiber kann eine Initialisierungssequenz an sein Gerät senden. Für Block-Gerätetreiber hat die Open-Funktion weniger praktische Bedeutung. Zu beachten ist, daß die Standard-Gerätetreiber von MS-DOS, nämlich »con«, »aux« und »prn« immer geöffnet sind und es auch bleiben.

Funktionscode 14: Geräte-Datei schließen. Die Close-Funktion ist erst ab MS-DOS 3.0 anwendbar, und zwar nur, wenn Bit 11 im Attributfeld des Gerät Headers eingeschaltet ist. Ein

zeichenorientierter Gerätetreiber kann eine Kontrollsequenz an sein Gerät senden. Für Block-Gerätetreiber hat die Close-Funktion weniger praktische Bedeutung. Zu beachten ist, daß die Standard-Gerätetreiber von MS-DOS, nämlich »con«, »aux« und »prn«, immer geöffnet sind und nie geschlossen werden.

Funktionscode 15: Speichermedium ist auswechselbar. Diese Funktion ist erst ab MS-DOS 3.0 anwendbar, und nur, wenn Bit 11 im Attributfeld des Device-Headers eingeschaltet ist. Sie ist lediglich für Block-Gerätetreiber bedeutsam. Ein Zeichen-Gerätetreiber hat ausschließlich das Erledigt-Bit zu setzen.

Im Statuswort wird nur das Beschäftigt-Bit verwendet. Eine »1« dort bedeutet, der Datenträger ist nicht auswechselbar, es handelt sich also zum Beispiel um eine Festplatte. Eine »0« bedeutet, der Datenträger, zum Beispiel eine Diskettenstation, kann gewechselt werden. Zu beachten ist, daß hierbei das Fehler-Bit von MS-DOS nicht ausgewertet wird.

Funktionscode 16: Daten fortgesetzt schreiben. Diese Funktion ist erst ab MS-DOS 3.0 anwendbar, und nur, wenn Bit 13 im Attributfeld des Gerät-Headers eingeschaltet ist. Sie ist ausschließlich für zeichenorientierte Gerätetreiber bedeutsam. Ein Block-Gerätetreiber hat lediglich das Erledigt-Bit zu setzen. Der Gerätetreiber überträgt so lange hintereinander Daten aus dem Übertragungspuffer des MS-DOS an sein Gerät, bis dieses meldet, es könne keine Daten mehr annehmen. Das kann zum Beispiel bei einem Drucker mit einem Puffer vorkommen. Der Drucker akzeptiert so lange Daten, bis sein interner Puffer gefüllt ist. Der Treiber hat die Zeichen gezählt, meldet sie MS-DOS, setzt den Status und gibt die Kontrolle zurück.

Soweit der Überblick über die einzelnen Funktionen, die ein Gerätetreiber beherrschen muß. Mit diesen Informationen können Sie nun endlich an die konkrete Programmierung eines einfachen Treibers gehen.

(M. Schmidt/ev)

```
[CR1] 1: MS_PRN PAGE 65,132
[YK3] 2: MS_PRN SEGMENT BYTE
[8T6] 3: ASSUME CS:MS_PRN, DS:MS_PRN, ES:N
      OTHING
[UD4] 4: ORG 0 ;Ein Device Driver
[K62] 5: ;beginnt bei 0
[P30] 6: ;
[FG2] 7: ; Device Header
[P30] 8: ;
[CF1] 9: DEV_HDR:
```

```
[LG2] 10: DW -1,-1 ;LINK
[TT3] 11: DW 0H ;ATTRIBUTES
[K95] 12: DW MS_STRAT ;STRATEGY ENTRY
[KH6] 13: DW MS_INT ;INTERRUPT ROUTINE ENTRY
[FE3] 14: DB " " ;DEVICE NAME
[P30] 15: ;
[SM2] 16: ; Befehlssteiler
[P30] 17: ;
[PL1] 18: COMM_TBL:
```

Listing eines allgemeinen Gerätetreiber-Rahmens


```

[D53] 19: DW MS_INIT ;INIT
[664] 20: DW MS_MEDIA ;MEDIA CHECK
[GV3] 21: DW MS_BBPB ;BUILD BPB
[F06] 22: DW MS_IOCTL ;IOCTL, NOT POSSIBLE
[TB3] 23: DW MS_READ ;INPUT
[XE6] 24: DW MS_NDREAD ;NON DESTRUCTIVE INPUT
[1U4] 25: DW MS_INSTAT ;INPUT STATUS
[CU4] 26: DW MS_INFLUSH ;FLUSH BUFFER
[ZW3] 27: DW MS_WRITE ;OUTPUT
[ZU5] 28: DW MS_VERWR ;OUTPUT MIT VERIFY
[645] 29: DW MS_OSTAT ;OUTPUT STATUS
[LY4] 30: DW MS_OFLUSH ;OUTPUT FLUSH
[K05] 31: DW MS_IOCTL ;IOCTL, OUTPUT
[8E4] 32: DW MS_DEVOP ;DEVICE OPEN
[7K4] 33: DW MS_DEVCL ;DEVICE CLOSE
[T35] 34: DW MS_RMEDIA ;REMOVABLE MEDIA
[9S5] 35: DW MS_OBUSY ;OUTPUT UNTIL BUSY
[P30] 36: ;
[7Z1] 37: ; VARIABLES
[P30] 38: ;
[7Z8] 39: R_PAK DD 00 ;Doppelwort, Adresse des Re
quest Headers
[P30] 40: ;
[WL4] 41: ; OFFSETS im Request Header
[P30] 42: ;
[7V3] 43: R_LEN = 0 ;LAENGENBYTE
[DC3] 44: R_UNIT = 1 ;SUBUNIT
[LU3] 45: R_CODE = 2 ;BEFEHLSCODE
[G44] 46: R_STAT = 3 ;STATUS-WORT
[G05] 47: R_MEDIA = 13 ;MEDIA DESCRIPTOR
[AV4] 48: R_TRANS = 14 ;TRANSFER ADDRESS
[577] 49: R_COUNT = 18 ;COUNT OF BLOCKS OR CHARACTERS
[HA6] 50: R_START = 20 ;FIRST BLOCK TO TRANSFER
[P30] 51: ;
[2D5] 52: ; STRATEGY Routine. Sichern der
[HP4] 53: ; Adresse des Request Headers
[P30] 54: ;
[F43] 55: MS_STRATP PROC FAR
[CP1] 56: MS_STRAT:
[7L5] 57: MOV WORD PTR [R_PAK],BX ;OFFSET
[T36] 58: MOV WORD PTR [R_PAK+2],ES ;SEGMENT
[3P0] 59: RET
[KH2] 60: MS_STRATP ENDP
[P30] 61: ;
[W56] 62: ; RETURN Routine. Jeder Rücksprung
[B84] 63: ; nach MS-DOS erfolgt hier.
[P30] 64: ;
[4Y2] 65: MS_RETP PROC FAR
[RY1] 66: MS_RETURN:
[8A1] 67: POP ES
[5A1] 68: POP DS
[L91] 69: POP BP
[X81] 70: POP DI
[AA1] 71: POP SI
[TA1] 72: POP DX
[PA1] 73: POP CX
[LA1] 74: POP BX
[GA1] 75: POP AX
[3P0] 76: RET
[V82] 77: MS_RETP ENDP
[P30] 78: ;
[XA6] 79: ; ENTRY-Routine. Jeder INTERRUPT-CALL
[RM5] 80: ; kommt hier an und wird über den
[5R6] 81: ; COMM_TBL gemäß dem Funktionscode auf
[LC6] 82: ; das für ihn bestimmte Label geleitet.
[P30] 83: ;
[SE1] 84: MS_INT:
[PC1] 85: PUSH AX
[TC1] 86: PUSH BX
[WC1] 87: PUSH CX
[ZC1] 88: PUSH DX
[GC1] 89: PUSH SI
[3B1] 90: PUSH DI
[TB1] 91: PUSH BP
[BC1] 92: PUSH DS
[EC1] 93: PUSH ES
[348] 94: LDS BX,CS:[R_PAK] ;DS:BX Adresse Request
Header
[P30] 95: ;
[5Z5] 96: ; Überprüfen des Funktionscodes,
[PB3] 97: ; darf nicht > 16 sein.
[P30] 98: ;
[488] 99: MOV AL,BYTE PTR DS:[BX].R_CODE ;Funktion
scode
[8N0] 100: CBW
[8C1] 101: PUSH CS
[5A1] 102: POP DS
[GH1] 103: CMP AL,16
[AS5] 104: JG CMD_ERR ;Funktionscode falsch
[PS2] 105: LEA SI,COMM_TBL
[HM1] 106: ADD SI,AX
[7G5] 107: ADD SI,AX ;zwei mal, Doppelwort
[BD8] 108: JMP WORD PTR DS:[SI] ;Verzweigen über CO
MM_TBL
[P30] 109: ;
[YE1] 110: CMD_ERR:

```

```

[P30] 111: ;
[MU4] 112: ; Fehlerhafter Funktionscode.
[P30] 113: ;
[027] 114: LDS BX,[R_PAK] ;Adresse Request Header
[P30] 115: ;
[K87] 116: ; ERROR-, DONE-, UNKNOWN-COMMAND-Bit setzen
[P30] 117: ;
[517] 118: MOV WORD PTR DS:[BX].R_STAT,100000010000
0011B
[LK2] 119: JMP MS_RETURN
[P30] 120: ;
[M75] 121: ; Hier folgen die Label für die
[WL4] 122: ; Codestrecken der einzelnen
[FP5] 123: ; Funktionscodes. Funktion 0 liegt
[YR5] 124: ; am Ende, ihr Hauptspeicherbereich
[9R5] 125: ; wird nach dem Booten freigegeben.
[P30] 126: ;
[XR5] 127: MS_MEDIA: ; Funktion 1: MEDIA CHECK
[LK2] 128: JMP MS_RETURN
[P30] 129: ;
[Y95] 130: MS_BBPB: ; Funktion 2: BUILD BPB
[LK2] 131: JMP MS_RETURN
[P30] 132: ;
[R55] 133: MS_IOCTL: ; Funktion 3: IOCTL
[LK2] 134: JMP MS_RETURN
[P30] 135: ;
[RT4] 136: MS_READ: ; Funktion 4: INPUT
[LK2] 137: JMP MS_RETURN
[P30] 138: ;
[XX7] 139: MS_NDREAD: ; Funktion 5: NON DESTRUCTIVE I
NPUT
[LK2] 140: JMP MS_RETURN
[P30] 141: ;
[CF6] 142: MS_INSTAT: ; Funktion 6: INPUT STATUS
[LK2] 143: JMP MS_RETURN
[P30] 144: ;
[2N6] 145: MS_INFLUSH: ; Funktion 7: FLUSH BUFFER
[LK2] 146: JMP MS_RETURN
[P30] 147: ;
[LF5] 148: MS_WRITE: ; Funktion 8: OUTPUT
[LK2] 149: JMP MS_RETURN
[P30] 150: ;
[SD7] 151: MS_VERWR: ; Funktion 9: OUTPUT MIT VERIFY
[LK2] 152: JMP MS_RETURN
[P30] 153: ;
[1N6] 154: MS_OSTAT: ; Funktion 10: OUTPUT STATUS
[LK2] 155: JMP MS_RETURN
[P30] 156: ;
[KE6] 157: MS_OFLUSH: ; Funktion 11: OUTPUT FLUSH
[LK2] 158: JMP MS_RETURN
[P30] 159: ;
[PR6] 160: MS_IOCTL: ; Funktion 12: IOCTL, OUTPUT
[LK2] 161: JMP MS_RETURN
[P30] 162: ;
[M16] 163: MS_DEVOP: ; Funktion 13: DEVICE OPEN
[LK2] 164: JMP MS_RETURN
[P30] 165: ;
[R46] 166: MS_DEVCL: ; Funktion 14: DEVICE CLOSE
[LK2] 167: JMP MS_RETURN
[P30] 168: ;
[XS6] 169: MS_RMEDIA: ; Funktion 15: REMOVABLE MEDIA
[LK2] 170: JMP MS_RETURN
[P30] 171: ;
[XB7] 172: MS_OBUSY: ; Funktion 16: OUTPUT UNTIL BUSY
[LK2] 173: JMP MS_RETURN
[P30] 174: ;
[553] 175: SCHLUSS LABEL WORD
[P30] 176: ;
[TK1] 177: MS_INIT:
[VK6] 178: ; Initialisierung. Dieser Code wird nur
[GA7] 179: ; einmal beim Laden des Drivers aufgerufen.
[9L6] 180: ; Danach wird er nicht mehr benötigt und
[A16] 181: ; ab dem Label SCHLUSS wird der Haupt-
[GT3] 182: ; speicher freigegeben.
[P30] 183: ;
[ES3] 184: ; Statuswort besetzen
[P30] 185: ;
[HX2] 186: LDS BX,[R_PAK]
[Y57] 187: MOV WORD PTR DS:[BX].R_STAT,0100H ;DONE
BIT
[P30] 188: ;
[X56] 189: ; Endadresse von Laufzeitcode setzen
[P30] 190: ;
[6T6] 191: LEA AX,SCHLUSS ;ab Label SCHLUSS
den
[7F8] 192: MOV DS:[BX].R_TRANS,AX ;Hauptspeicher fr
eigeben
[V53] 193: MOV DS:[BX+16],CS
[LK2] 194: JMP MS_RETURN
[P30] 195: ;
[632] 196: MS_PRN ENDS
[P30] 197: ;
[4M0] 198: END

```

Listing eines allgemeinen Gerätetreiber-Rahmens

Ein Druckertreiber im Selbstbau

In den beiden vorhergehenden Artikeln haben Sie die theoretischen Hintergründe von Gerätetreibern unter MS-DOS kennengelernt. Nun gelangen Sie endlich ans Ziel und programmieren den Code eines funktionierenden Gerätetreibers. Dabei wird Ihnen die vorangegangene »trockene« Theorie gute Dienste leisten.

In Listing 1 finden Sie einen vollständigen und funktionierenden Gerätetreiber. Er führt eine Druckeremulation durch. Die Portadresse und die logische Gerätebezeichnung können Sie wählen, das heißt, Sie können entscheiden, ob der Treiber als »lpt1«, »lpt2« oder »lpt3« übernommen wird. Dazu müssen Sie den Eintrag des »Device Name« im Device-Header an der Marke »dev_hdr« entsprechend ändern. Danach muß der Treiber generiert werden. Sie können weiterhin unabhängig vom Namen des Geräts entscheiden, ob er über die erste, zweite oder dritte Drucker-schnittstelle (Druckerkarte) ausgegeben soll. Diese Angaben werden in der Aufrufzeile der Datei »config.sys« gemacht. Dann haben Sie noch die Möglichkeit, die Emulation zu bestimmen. Die Umwandlungstabelle nämlich können Sie nach den Spezifikationen Ihres Druckers erstellen. Allerdings ist in diesem Code nur eine Emulation für eine Eins-zu-Eins-Zuordnung vorgesehen; Sie können also nur jeweils ein Zeichen durch genau ein anderes ersetzen. Wenn Sie den Code aber einmal verstanden haben, läßt sich sehr leicht auch eine andere Zuordnung programmieren, zum Beispiel, um den Drucker bei bestimmten Zeichen in den Grafikmodus zu schalten und nicht vorhandene Zeichen als Grafik zu erzeugen. Die Möglichkeiten, die Ihnen hier zur Verfügung stehen, gehen so weit, daß Sie Ihren Drucker als kompatiblen Standarddrucker betreiben könnten, obwohl der Drucker selbst gar nicht kompatibel ist.

Die Fehlersuche kann bei der Programmierung von Gerätetreibern sehr problematisch werden

Bevor Sie sich mit der Programmierung befassen, müssen Sie noch wissen, wie ein Gerätetreiber compiliert und in das Betriebssystem aufgenommen wird. Es ist ebenfalls recht informativ, einige Hinweise darüber zu erhalten, wie man »im Falle eines Falles« die Fehlersuche organisiert. Bei Gerätetreibern gestaltet sich dies

nämlich nicht so einfach wie bei »normalen« Anwendungsprogrammen.

Der Treiber in Listing 1 ist mit dem Makro-Assembler »MASM«, Version 4.0 von Microsoft, assembliert worden. Der Code beginnt bei »org 0« und enthält keine Include-Anweisungen. Die Assemblierung ist somit unproblematisch. Der Treiber hat den Dateinamen »msprn« und wird mit folgender Anweisung übersetzt und gebunden:

```
masm msprn.asm,msprn.obj,msprn.lst,nul
link msprn,msprn,nul,nul
```

Dabei erhalten Sie ein Exe-Modul mit dem Namen »msprn«. Das Exe-Modul ist allerdings noch nicht die fertige Treiber-Version. Wie Sie sicherlich wissen, haben Gerätetreiber die Datei-Erweiterungen »sys« oder »bin«. Dies ist nicht nur eine bloße Namensvergabe. Das Format der Datei hat bestimmten Anforderungen zu genügen. Die Umwandlung erfolgt mit der MS-DOS-Routine »exe2bin«. Folgende Anweisung schließlich liefert Ihnen den fertigen Treiber:

```
exe2bin msprn.exe msprn.sys
```

Nun sind Sie beinahe am Ziel. Das einzige, was noch zu tun bleibt, ist, den Treiber in die Datei »config.sys« einzutragen. Dies geschieht durch die Anweisung

```
device = msprn.sys /0,0
```

Danach müssen Sie Ihren PC neu starten. Wenn Sie das Listing richtig abgetippt haben, wird sich der Treiber »msprn« mit einer informativen Meldung auf dem Bildschirm bemerkbar machen und Ihnen mitteilen, daß die Installation erfolgreich war. Wenn Sie eine Testdatei »test.dat« mit Umlauten schreiben und die beiden folgenden Befehle eingeben, werden Sie feststellen, daß der Treiber aktiv ist:

```
copy test.dat lpt1
```

```
copy test.dat lpt2
```

Auf dem Drucker erscheinen die Umlaute beide Male umgewandelt. Dies kommt daher, daß der Gerätetreiber »msprn« durch die beiden Parameter der Zeile

```
device = msprn /0,0
```

auf die erste Drucker-schnittstelle gelegt ist. Der erste Parameter legt die

Druckerportadresse des logischen Geräts »lpt2« fest und kann die Werte von 0 bis 2 annehmen. Wenn Sie diesen Parameter auf 1 setzten, würde der Befehl:

```
copy test.dat lpt2
```

auf die zweite Druckerkarte wirken, die dann auch vorhanden sein sollte. Der zweite Parameter legt die Drucker-schnittstelle für den Drucker-Interrupt 17hex fest. Er kann ebenfalls die Werte 0 bis 2 annehmen. MS-DOS druckt generell über den Drucker 0.

Parameter steuern alles – ein Fehler hierbei führt fast immer zu langem Suchen

Wenn Sie also den zweiten Parameter auf 1 setzten, würde zunächst nichts geschehen. Es würde erst dann eine Emulation und eine Ansprache des zweiten Druckerports stattfinden, wenn Sie ein Anwendungsprogramm schreiben, das den Interrupt 17hex direkt anspricht und den Drucker 1 verlangt. Dann allerdings sollte auch eine zweite Druckerkarte vorhanden sein. Um nun zu zeigen, daß der Gerätetreiber »lpt1« nicht, »lpt2« dagegen sehr wohl emuliert, schreiben Sie folgenden Aufruf in die Datei »config.sys«:

```
device = msprn.sys /0,1
```

Beachten Sie, daß die Leerstelle zwischen ».sys« und dem Schrägstrich zwingend vorhanden sein muß. Nach der entsprechenden Änderung von »config.sys« starten Sie Ihren Rechner erneut. Wenn Sie jetzt die Befehle:

```
copy test.dat lpt1
```

```
copy test.dat lpt2
```

eingeben, wird offensichtlich, daß die Emulation einmal ausgeführt wurde, das andere Mal dagegen nicht. Das ist richtig und zeigt, daß Ihr selbstgeschriebener Treiber gut funktioniert.

Was aber, wenn Sie ihn weiterentwickeln wollen, wenn Sie Änderungen im Code angebracht haben, und er läuft dann zuerst einmal gar nicht – ein Umstand, der ja ganz »normal« ist. Haben Sie beim »Spielen« mit »msprn« schon einmal versucht, ihn im Hauptspeicher zu finden? Dies ist zunächst einmal gar nicht so einfach, weil er nämlich in einem Bereich des Betriebssystems liegt. Wie auch immer, um die Fehlersuche, zumindest in einer rudimentären Form, überhaupt möglich zu machen, müssen Sie Ihren selbstgeschriebenen Treiber finden. Im Fall von »msprn« ist es unproblematisch: Er »verbiegt« nämlich den Drucker-Interrupt 17hex auf eine eigene Routine. Somit enthalten die von diesem Interrupt-Vektor belegten Speicherzellen »0000:005C«

die Adresse der Marke »ms__printx«. Damit können Sie durch einfaches Auslesen dieser Speicherzellen (zum Beispiel mit »debug«) die Speicheradresse bestimmen, an der der Treiber abgelegt ist.

Die Fehlersuche selbst ist eine andere Sache. Mit »debug« an die Sache heranzugehen, ist bei einem Gerätetreiber ziemlich schwierig. Sie können natürlich alle Parameter so setzen, wie MS-DOS dies tun würde, und den Treiber dann mit »debug« im Einzelschrittmodus durchgehen. Diese Methode versagt jedoch dann, wenn sich der Treiber gleich beim Starten des Betriebssystems unangenehm bemerkbar macht und der Computer einfach stehenbleibt. Aus diesem Grunde sollten Sie derartige Tests niemals von der Festplatte »fahren«. Stellen Sie sich für solche Zwecke lieber eine Startdiskette zusammen, mit der Sie ausgiebig experimentieren können. Geht dann nämlich etwas schief, ziehen Sie einfach die Diskette aus dem Laufwerk und starten den Rechner neu von der Festplatte, auf der sich natürlich eine funktionierende Config-Datei befindet.

Das Listing des Gerätetreibers ist sehr übersichtlich aufgebaut

Doch wenden Sie sich nun Listing 1 zu. Die ersten beiden Definitionen beschreiben das Link-Feld, welches auf »-1« gesetzt ist, da nur ein einziger Gerätetreiber im Listing enthalten ist. Insofern existiert keine Verketzung zum nächsten Treiber. In den Attributen wird nur angegeben, daß es sich um einen zeichenorientierten Gerätetreiber handelt. Das geschieht durch Setzen von Bit 15. Die Einsprungspunkte für die Strategy- und die Interrupt-Routine liegen auf den Marken »ms__strat« beziehungsweise »ms__int«. Der logische Name des Treibers ist »lpt2«.

Gemäß dem Attributwort sind die Funktionen »init«, »output«, »output with verify«, »output status« und »output flush« erlaubt. Sollte ein anderer Funktionscode ankommen, wird zur Marke »ms__err1« verzweigt. Der Befehlsteiler sorgt für die korrekte Ausführung der einzelnen Routinen für jeden Funktionscode. In ihm sind 17 Einträge vorhanden. Sie sind in der Reihenfolge ihrer Befehls-codes angeordnet. Der Vollständigkeit halber muß erwähnt werden, daß noch zwei weitere Funktionen existieren. Dies sind »Generic IOCTL Request« und »Get/Set Logical Drive Map«. Beide sind erst ab MS-DOS 3.2 verfügbar und für den vorliegenden Gerätetreiber nicht relevant, da in

den Attributen ein IOCTL-Aufruf nicht zugelassen ist.

Die Druckertabelle in Listing 1 ist für einen OKI-MicrolineAEDrucker konzipiert, funktioniert so aber auch für viele Drucker, die nur den Standard-ASCII-Zeichensatz kennen und keine Umlaute ausgeben wollen. Wenn Sie diese Tabelle zunächst einmal verwenden wollen, dann beachten Sie, daß der Steuercode 15hex in 1Dhex umgesetzt wird. Dadurch kann der MicrolineAEDrucker durch die Mode-Anweisung von MS-DOS in den »Compressed Mode« umgeschaltet werden. Das sieht bei Ihrem Drucker unter Umständen anders aus. Weiterhin könnte es bei Ihrem Drucker notwendig werden, alle Zeichen ab 80hex in Leerzeichen umzuwandeln. In diesem Fall tragen Sie in die Druckertabelle ab dem Byte 80hex bis FFhex den Code 20hex ein.

Dabei wird deutlich, wie der Treiber die Umwandlung vornimmt. Ein kompatibler Code wird von MS-DOS an den Treiber gesendet, zum Beispiel 94hex, das ist der Buchstabe »ö« im IBM-Zeichensatz. Der Treiber holt sich nun den Code, der an der Position 94hex in der Druckertabelle steht, also 7Chex, und sendet diesen an den Drucker. Das funktioniert in dieser Form auch, wenn der Drucker-Interrupt 17hex direkt angesprochen wird, da der Treiber einen Interrupt-Handler für den Interrupt 17hex aufsetzt.

Wird der Treiber angesprochen, so führt MS-DOS als erstes einen »Far Call« auf die Marke »ms__strat« aus. Dort rettet der Gerätetreiber die Adresse des Request-Headers und gibt die Kontrolle sofort wieder an das Betriebssystem zurück. Gleich danach führt MS-DOS einen weiteren »Far Call« auf die Marke »ms__int« aus. Der Treiber rettet die Register und überprüft den Funktionscode. Mit dessen Hilfe berechnet der Treiber sich den Einsprungspunkt in den Befehlsteiler, um durch einen indirekten Sprung an die für den Funktionscode bestimmte Marke zu kommen. Dort wird die Funktion ausgeführt und zur Marke »ms__return« zurück gesprungen. »ms__return« stellt die Register wieder her und gibt die Kontrolle über einen »Far Return« an MS-DOS zurück.

An der Marke »ms__printx« finden Sie den Interrupt-Handler für den Interrupt 17hex. Die Variable »iport« wird bei Initialisierung des Treibers gemäß der Aufrufzeile in der Datei »config.sys« belegt. Der Interrupt-Handler emuliert nur dann, wenn der in Register DX angegebene Druckerport mit »iport« übereinstimmt. Ansonsten wird das Zeichen ohne Emulation gedruckt.

Beim Neustart führt der Treiber die

Funktion »init«, (Funktionscode 0) aus. Nach dem Befehlsteiler springt er an die Marke »ms__init«. Zuerst wird der Cursor gesetzt und eine Identifikationsmeldung auf den Bildschirm gedruckt. Danach werden die Parameter der Aufrufzeile der Datei »config.sys« kontrolliert. Wenn diese nicht der Syntaxvorschrift entsprechen, verzweigt der Treiber zur Fehlerbehandlung bei der Marke »f__ret«. Dort setzt er den von ihm benötigten Hauptspeicherbereich auf »0« und teilt MS-DOS dadurch mit, daß er nicht geladen ist.

Nach einer Fehlermeldung hat das Betriebssystem wieder die vollständige Kontrolle

Nach einer Fehlermeldung auf dem Bildschirm übergibt er die Kontrolle an das Betriebssystem. Wenn die Syntaxvorschriften eingehalten worden sind, belegt der Treiber die Variable »lpt2« mit dem des ersten und die Variable »iport« mit dem Wert des zweiten Parameters. Der Drucker wird initialisiert und der Interrupt-Handler auf den Interrupt 17hex aufgesetzt. Der Original-Vektor 17hex wird auf eine neue Interrupt-Adresse gelegt, die in einer EQU-Anweisung mit der Bezeichnung »int__17« am Anfang des Listings definiert ist. Sie kann selbstverständlich von Ihnen anders gewählt werden.

Die Funktionen 8 und 9 (»output« und »output with verify«) werden vom Befehlsteiler auf die gemeinsame Routine »ms__print« geleitet. Die Funktion »output with verify« wird von MS-DOS dann ausgewählt, wenn das Verify-Flag gesetzt ist, was durch den MS-DOS-Befehl

verify on

festgelegt werden kann. Für einen Drucker ist aber eine Verifizierung der Ausgabe nicht möglich. Aus diesem Grunde sind beide Ausgabefunktionen identisch. In der Routine »ms__print« greift der Treiber auf die Adresse des Übertragungspuffers (»Transfer Area«) zu, die im Request-Header steht. Hier findet er das zu druckende Zeichen. Im Falle des vorliegenden Listings wird MS-DOS immer nur ein Zeichen liefern, weil das Bit für »output until busy« in den Attributen nicht eingeschaltet ist. Der Treiber liest das zu druckende Zeichen und verzweigt in das Unterprogramm »pr__stat«. Diese Routine gibt die Kontrolle erst dann wieder zurück, wenn der Drucker bereit ist. Der Treiber übersetzt das zu druckende Zeichen mit Hilfe der Druckerumwandlungstabelle »pr__tab«, lädt den Druckerport aus der Variablen »lpt2«

und schickt das umgewandelte Zeichen im Unterprogramm »prnt_out« auf den Drucker. Danach muß er die Rückmeldungen, den Status und die Anzahl gedruckter Zeichen in den Request-Header eintragen. Die Anzahl verarbeiteter Zeichen ist hier natürlich immer 1. Über die Marke »ms_return« wird die Kontrolle an MS-DOS zurückgegeben.

Die Funktion 10 (»output status«) wird auf die Routine »ms_sta« geführt. Der Treiber lädt den Druckerport und fragt den Status des Druckers ab. Ausgewertet wird hierbei nur das Busy-Bit des Druckers. Ist der Drucker beschäftigt, setzt der Treiber im Request-Header das Busy-Bit, anderenfalls das Erledigt-Bit. Im Falle des hier besprochenen Treibers wird diese Funktion nicht besonders bedeutsam sein, da die Funktion »output until busy« nicht implementiert ist. Insofern wird eine Statusabfrage wohl in jedem Fall ein »ready« erbringen.

Für die Druckerabfrage wurde hier die Funktion 02hex des BIOS-Interrupt 17hex verwendet. Dies ist bei dem Oki-MicrolineAEDrucker möglich. Wenn Sie aber einen Drucker besitzen, bei dem der Status ganz an-

ders abgefragt wird, dann müssen Sie unter Umständen diesen Listingteil anders programmieren.

Die nicht erlaubten Funktionen werden alle unter der Marke »ms_err1« abgehandelt. Nach einer Fehlermeldung wird das Fehler-Bit gesetzt und der Fehlercode für »unknown command« in den Request-Header eingetragen.

Nicht erlaubte Funktionen werden unter der Marke »ms_err1« behandelt

Die Funktion 11 (»output flush«) wird unter der Marke »ms_oflush« abgehandelt. Dabei wird eine eventuell vorhandene Ausgabe vernichtet. Im Falle dieses Gerätetreibers dürfte kaum eine Ausgabe zum Vernichten vorhanden sein, da er ja keinen eigenen Puffer besitzt und nur jeweils ein Zeichen von MS-DOS gesendet bekommt. Zur Funktion 11 könnten Schwierigkeiten mit dem Drucker geführt haben, und aus diesem Grunde wird er initialisiert. Das Erledigt-Bit im Request-Header wird gesetzt, und über die Marke »ms_return« erhält MS-DOS die Kontrolle zurück. Auch hier, wie bei der Funktion »init«,

wird die Initialisierung des Druckers über die Funktion 02hex des BIOS-Interrupt 017hex durchgeführt. Falls Ihr Drucker anders initialisiert wird, muß an dieser Stelle der entsprechende Code eingetragen werden. Der Interrupt-Handler für den BIOS-Interrupt 17hex liegt an der Marke »ms_printx«. Hierhin wird nicht über den Befehlsteiler verzweigt, sondern über den Befehl

int 17h

eines Anwenderprogramms oder des MS-DOS. Der Interrupt-Handler erhält das zu druckende Zeichen im Register AL und den von der rufenden Routine verlangten Druckerport im Register DX. Stimmt DX nicht mit der Variablen »iport« überein, so soll ein anderer Drucker bedient werden, und es soll auch keine Emulation stattfinden. Der Interrupt-Handler gibt in diesem Fall die Kontrolle an den ursprünglichen Interrupt 17hex ab. Das gleiche geschieht, wenn eine andere BIOS-Funktion als »print character« im Register AH angegeben ist. Wenn das Zeichen bearbeitet werden muß, dann stimmt die Angabe des verlangten Druckers im Register DX mit dem zweiten Parameter der Treiber-Aufrufzeile der Datei »config.sys« über-

Wissen Sie mehr von PCs als andere Leute?

Dann schreiben Sie uns. Die Redaktion DOS International sucht aktive Mitarbeiter, die über interessante Gebiete aus der »Computerei« berichten wollen. MS-DOS, DOS Plus, GEM, Drucker, Basteleien – es gibt nichts, worüber nicht andere Leser unserer Zeitschrift auch etwas lesen wollen.

Ihren Einsatz lassen wir uns natürlich etwas kosten. Ihr Manuskript wird selbstverständlich honoriert. Bilder, Zeichnungen, Tabellen und Listings werden gesondert berechnet.

Doch bevor Sie sich hinsetzen und mit dem Schreiben beginnen:

Rufen Sie uns an.

Sagen Sie uns, was Sie schreiben wollen, und wir klären, ob – und wann – der Artikel in unser Konzept paßt.

Der Artikel muß mit einem Textsystem erfaßt werden. Wir brauchen von Ihnen eine Diskette im MS-DOS-Format, auf der der Text als reine ASCII-Datei steht.

Schreiben Sie uns auch, wenn Sie nur Vorschläge für Themen haben.

DOS International ist nämlich eine Zeitschrift von aktiven PC-Besitzern für aktive PC-Benutzer. Besonders das DOS-Forum lebt von Ihrer Mitarbeit. Auch die Redaktion kann nicht alle Antworten wissen. Fragen an uns, die nicht sofort beantwortet werden können, werden im DOS-Forum veröffentlicht. Wissen Sie eine Antwort, so wird diese veröffentlicht.

Sie sehen: DOS International ist die PC-Zeitschrift, in der das Wissen der PC-Benutzer untereinander ausgetauscht wird.

**Redaktionsbüro Everts & Hagedorn · Redaktion »DOS International«
Hans-Stießberger-Str. 3 · 8013 Haar/München · Tel.: (0 89) 46 10 01-0**

ein. Der Interrupt-Handler wandelt das Zeichen mit Hilfe der Druckenumwandlungstabelle um und sendet es in dem Unterprogramm »prnt_out« an den Drucker. Der Rücksprung in die rufende Routine erfolgt in diesem Fall natürlich über einen Iret-Befehl.

Damit wäre das Thema der Programmierung von Gerätetreibern unter MS-DOS grundsätzlich abgehandelt. Es ist sicher deutlich geworden, daß es sich bei einem Gerätetreiber auch »nur« um ein Programm handelt. Al-

lerdings erfordert es seitens des Programmierers eine ganze Menge mehr Disziplin als andere Anwendungsprogramme. Man denke in diesem Zusammenhang nur an die eingeschränkten Möglichkeiten zur Fehlersuche. Auf der anderen Seite ist die Struktur eines Gerätetreibers durch die Besonderheit seines Aufrufes durch MS-DOS ziemlich fest vorgegeben. Weiterhin sind die Funktionen und die Art der Parameterübergabe eindeutig dokumentiert. Mit dem vorliegenden Listing kann, mit den

entsprechenden Erweiterungen, so ziemlich jeder Drucker betrieben werden. Das geht so weit, daß in einer Textsoftware der Treiber für einen kompatiblen Drucker gewählt, aber ein beliebiger und inkompatibler betrieben wird – selbstverständlich mit allen Sonderfunktionen wie Fettdruck und so weiter, die der Kompatible zuläßt. Einschränkungen entstehen nur dort, wo die Drucker-Hardware auch mit Software-Unterstützung nicht mehr weiter kann.

(M. Schmidt/ev)

```
[CR1] 1: PAGE 65,132
[LL3] 2: MS_PRN SEGMENT BYTE
[HS6] 3: ASSUME CS:MS_PRN, DS:NOTHING, ES:N
      NOTHING
[P30] 4: ;
[X15] 5: ;=====
[RP4] 6: ; M. Schmidt, Speyererstr. 23,
[SL2] 7: ; 60000 Frankfurt 1
[X15] 8: ;=====
[P30] 9: ;
[X15] 10: ;=====
[BW3] 11: ; DEVICE=MSPRN.SYS /x,y
[6H4] 12: ; x = Printer Port für LPT2
[UL4] 13: ; y = Printer Port für INT 17
[VB3] 14: ; Version 1.0 / 16.7.1988
[ES5] 15: ; modifiziert f. DOS International
[X15] 16: ;=====
[P30] 17: ;
[2B2] 18: ;Device Header
[P30] 19: ;
[CF1] 20: DEV_HDR:
[PC2] 21: DW -1,-1 ;LINK
[VL3] 22: DW 8000H ;ATTRIBUTES
[G25] 23: DW MS_STRAT ;STRATEGY ENTRY
[D25] 24: DW MS_INT ;INTERRUPT ENTRY
[HT3] 25: DB "LPT2" ;DEVICE NAME
[P30] 26: ;
[2F2] 27: ;Befehlssteiler
[P30] 28: ;
[PL1] 29: COMM_TBL:
[C03] 30: DW MS_INIT ;INIT
[C14] 31: DW MS_ERR1 ;MEDIA CHECK
[CL3] 32: DW MS_ERR1 ;BUILD BPB
[C13] 33: DW MS_ERR1 ;IOCTL
[E33] 34: DW MS_ERR1 ;INPUT
[H06] 35: DW MS_ERR1 ;NON DESTRUCTIVE INPUT
[LC4] 36: DW MS_ERR1 ;INPUT STATUS
[X64] 37: DW MS_ERR1 ;FLUSH BUFFER
[TP3] 38: DW MS_PRINT ;OUTPUT
[6D5] 39: DW MS_PRINT ;OUTPUT MIT VERIFY
[6G4] 40: DW MS_STA ;OUTPUT STATUS
[TR4] 41: DW MS_OFLUSH ;OUTPUT FLUSH
[2B4] 42: DW MS_ERR1 ;IOCTL, OUTPUT
[S34] 43: DW MS_ERR1 ;DEVICE OPEN
[V64] 44: DW MS_ERR1 ;DEVICE CLOSE
[ST4] 45: DW MS_ERR1 ;REMOVABLE MEDIA
[NG5] 46: DW MS_PRINT ;OUTPUT UNTIL BUSY
[P30] 47: ;
[Z23] 48: ;Variable und Meldungen
[P30] 49: ;
[VM4] 50: R_PAK DD 00 ;Request Header
[M15] 51: CURP2 DW 0000H ;Cursor Position
[0R4] 52: INT_17 EQU 078H ;neuer INT 17
[P30] 53: ;
[EL4] 54: ;OFFSETS im Request Header
[P30] 55: ;
[KT3] 56: R_LEN = 0 ;Längenbyte
[DC3] 57: R_UNIT = 1 ;Subunit
[LU3] 58: R_CODE = 2 ;Befehlscode
[G44] 59: R_STAT = 3 ;Status-Wort
[G05] 60: R_MEDIA = 13 ;MEDIA DESCRIPTOR
[AV4] 61: R_TRANS = 14 ;TRANSFER ADDRESS
[DP6] 62: R_COUNT = 18 ;Anzahl Zeichen oder Blöcke
```

```
[727] 63: R_START = 20 ;Anfang Übertragungsbereich
[P30] 64: ;
[505] 65: MSG_1 DB "M S P R N Version 1.0, "
[RX4] 66: DB "von M. Schmidt, Frankfurt, "
[0M4] 67: DB "für DOS International"
[9U1] 68: MSG_1E EQU $
[355] 69: MSG_2 DB "M S P R N Version 1.0, "
[9H5] 70: DB "Initialisierung erfolgreich "
[BU1] 71: MSG_2E EQU $
[905] 72: MSG_3 DB "M S P R N Version 1.0, "
[MY5] 73: DB "Treiber wird nicht installiert"
[DU1] 74: MSG_3E EQU $
[D05] 75: MSG_5 DB "M S P R N Version 1.0, "
[2R4] 76: DB "unbekannter Befehlscode"
[HU1] 77: MSG_5E EQU $
[F05] 78: MSG_6 DB "M S P R N Version 1.0, "
[V93] 79: DB "Laufzeitfehler, "
[DS4] 80: DB "unerlaubter Befehlscode"
[LU1] 81: MSG_6E EQU $
[P30] 82: ;
[X25] 83: LPT2 DW 00 ;Logischer Drucker
[MN5] 84: IPORT DW 00 ;DRUCKER PORT für INT 17
[P30] 85: ;
[4P0] 86: PAGE
[P30] 87: ;
[HS2] 88: ;Drucker Tabelle
[P30] 89: ;
[AF2] 90: PR_ID DB "OKI-92"
[W35] 91: PR_TAB DB 0H, 1H, 02H, 003H, 004H, 005H
[9M4] 92: DB 006H, 007H, 008H, 009H, 00AH, 00BH
[NN4] 93: DB 00CH, 00DH, 00EH, 01DH, 010H, 011H
[8M4] 94: DB 01EH, 013H, 014H, 015H, 016H, 017H
[FP4] 95: DB 018H, 019H, 01AH, 01BH, 01CH, 01DH
[7M4] 96: DB 01EH, 01FH, 020H, 021H, 022H, 023H
[TL4] 97: DB 024H, 025H, 026H, 027H, 028H, 029H
[NS4] 98: DB 02AH, 02BH, 02CH, 02DH, 02EH, 02FH
[VK4] 99: DB 030H, 031H, 032H, 033H, 034H, 035H
[ZN4] 100: DB 036H, 037H, 038H, 039H, 03AH, 03BH
[9R4] 101: DB 03CH, 03DH, 03EH, 03FH, 040H, 041H
[5M4] 102: DB 042H, 043H, 044H, 045H, 046H, 047H
[5S4] 103: DB 048H, 049H, 04AH, 04BH, 04CH, 04DH
[XN4] 104: DB 04EH, 04FH, 050H, 051H, 052H, 053H
[FN4] 105: DB 054H, 055H, 056H, 057H, 058H, 059H
[BU4] 106: DB 05AH, 05BH, 05CH, 05DH, 05EH, 05FH
[HM4] 107: DB 060H, 061H, 062H, 063H, 064H, 065H
[NR4] 108: DB 066H, 067H, 068H, 069H, 06AH, 06BH
[ZS4] 109: DB 06CH, 06DH, 06EH, 06FH, 070H, 071H
[VN4] 110: DB 072H, 073H, 074H, 075H, 076H, 077H
[VT4] 111: DB 078H, 079H, 07AH, 07BH, 07CH, 07DH
[2S4] 112: DB 07EH, 07FH, 080H, 081H, 082H, 083H
[DS4] 113: DB 084H, 085H, 086H, 087H, 088H, 089H
[KV4] 114: DB 08AH, 08BH, 08CH, 08DH, 08EH, 08FH
[GR4] 115: DB 090H, 091H, 092H, 093H, 094H, 095H
[6T4] 116: DB 096H, 097H, 098H, 099H, 09AH, 09BH
[ZV4] 117: DB 09CH, 09DH, 09EH, 09FH, 0A0H, 0A1H
[FV4] 118: DB 0A2H, 0A3H, 0A4H, 0A5H, 0A6H, 0A7H
[FZ4] 119: DB 0A8H, 0A9H, 0AAH, 0ABH, 0ACH, 0ADH
[7X4] 120: DB 0AEH, 0AFH, 0B0H, 0B1H, 0B2H, 0B3H
[TW4] 121: DB 0B4H, 0B5H, 0B6H, 0B7H, 0B8H, 0B9H
[N15] 122: DB 0BAH, 0BBH, 0BCH, 0BDH, 0BEH, 0BFH
[VV4] 123: DB 0C0H, 0C1H, 0C2H, 0C3H, 0C4H, 0C5H
[ZY4] 124: DB 0C6H, 0C7H, 0C8H, 0C9H, 0CAH, 0CBH
[905] 125: DB 0CCH, 0CDH, 0CEH, 0CFH, 0D0H, 0D1H
```



```

[5X4] 126: DB 0D2H, 0D3H, 0D4H, 0D5H, 0D6H, 0D7H
[515] 127: DB 0D8H, 0D9H, 0DAH, 0DBH, 0DCH, 0DDH
[TX4] 128: DB 0DEH, 0DFH, 0E0H, 0E1H, 0E2H, 0E3H
[FY4] 129: DB 0E4H, 0E5H, 0E6H, 0E7H, 0E8H, 0E9H
[B35] 130: DB 0EAH, 0EBH, 0ECH, 0EDH, 0EEH, 0EFH
[HX4] 131: DB 0F0H, 0F1H, 0F2H, 0F3H, 0F4H, 0F5H
[N05] 132: DB 0F6H, 0F7H, 0F8H, 0F9H, 0FAH, 0FBH
[MF3] 133: DB 0FCH, 0FDH, 0FEH, 0FFH
[P30] 134: ;
[2D5] 135: ; STRATEGY Routine. Sichern der
[HP4] 136: ; Adresse des Request Headers
[P30] 137: ;
[F43] 138: MS_STRATP PROC FAR
[CP1] 139: MS_STRAT:
[BU5] 140: MOV WORD PTR [R_PAK],BX ;OFFSET
[X56] 141: MOV WORD PTR [R_PAK+2],ES ;SEGMENT
[3P0] 142: RET
[KH2] 143: MS_STRATP ENDP
[P30] 144: ;
[NZ6] 145: ; RETURN Routine. Jeder Rücksprung nach
[W84] 146: ; MS-DOS erfolgt von hier.
[P30] 147: ;
[4Y2] 148: MS_RETP PROC FAR
[RY1] 149: MS_RETURN:
[8A1] 150: POP ES
[5A1] 151: POP DS
[L91] 152: POP BP
[X81] 153: POP DI
[AA1] 154: POP SI
[TA1] 155: POP DX
[PA1] 156: POP CX
[LA1] 157: POP BX
[GAI] 158: POP AX
[3P0] 159: RET
[V82] 160: MS_RETP ENDP
[P30] 161: ;
[TY6] 162: ; Interrupt-Handler für INT 17 zu INT_17
[P30] 163: ;
[8Y1] 164: MS_PRINTX:
[BC1] 165: PUSH DS
[8C1] 166: PUSH CS
[5A1] 167: POP DS
[AD5] 168: CMP DX,IPORT ;DX = PRINTER PORT
[7B4] 169: JNZ MS_PRINT1 ;nicht LPT2
[P30] 170: ;
[S26] 171: ; LPT2 ist gewählt, welche Funktion?
[P30] 172: ;
[CT1] 173: CMP AH,00H
[8F5] 174: JNZ MS_PRINT1 ;nicht PRINT-CHAR
[P30] 175: ;
[UG6] 176: ; PRINT-CHARACTER-Funktion, emulieren,
[333] 177: ; Zeichen ist in AL
[P30] 178: ;
[WC1] 179: PUSH CX
[EC1] 180: PUSH ES
[GC1] 181: PUSH SI
[3B1] 182: PUSH DI
[TC1] 183: PUSH BX
[KS1] 184: MOV SI,AX
[AN6] 185: MOV AL,PR_TAB [SI] ;Zeichen umwandeln
[XE6] 186: CALL PRNT_OUT ;an Drucker schicken
[LA1] 187: POP BX
[X81] 188: POP DI
[AA1] 189: POP SI
[8A1] 190: POP ES
[PA1] 191: POP CX
[3K3] 192: JMP SHORT MS_PRINT2
[UUV1] 193: MS_PRINT1:
[0Y1] 194: INT INT_17
[WV1] 195: MS_PRINT2:
[5A1] 196: POP DS
[GR0] 197: IRET
[P30] 198: ;
[4P0] 199: PAGE
[P30] 200: ;
[PR2] 201: SETCUR PROC NEAR
[P30] 202: ;
[U91] 203: ; ENTRY
[7H3] 204: ; Setzen des Cursors,
[W84] 205: ; zerstörte Register : AX
[L73] 206: ; Eingabeparameter:
[WK2] 207: ; BH: PAGE-NUMBER
[BR3] 208: ; DH: LINE-NUMBER (0 - 24d)
[684] 209: ; DL: COLUMN-NUMBER (0 - 79d)
[P30] 210: ;
[BH1] 211: SET_CUR:
[P30] 212: ;
[8D6] 213: MOV CURP2,DX ;sichern Cursor Position
[0X5] 214: MOV AH,02H ;Funktion: SET-CURSOR
[0A1] 215: INT 010H

```

```

[3P0] 216: RET
[N02] 217: SETCUR ENDP
[P30] 218: ;
[YP2] 219: GETCUR PROC NEAR
[P30] 220: ;
[WN4] 221: ; Lesen der Cursorposition
[B85] 222: ; zerstörte Register : AX, CX, DX
[L73] 223: ; Eingabeparameter:
[WK2] 224: ; BH: PAGE-NUMBER
[PB3] 225: ; Returnparameter :
[GU3] 226: ; CH: CURSOR START LINE
[T93] 227: ; CL: CURSOR END LINE
[534] 228: ; DH: LINE POSITION (0 - 24d)
[HB4] 229: ; DL: COLUMN POSITION (0 - 79)
[P30] 230: ;
[LG1] 231: GET_CUR:
[P30] 232: ;
[ZV5] 233: MOV AH,03H ;Funktion: GET-CURSOR
[0A1] 234: INT 010H
[8D6] 235: MOV CURP2,DX ;Sichern Cursor Position
[3P0] 236: RET
[XZ1] 237: GETCUR ENDP
[P30] 238: ;
[K83] 239: PR_STATP PROC NEAR
[P30] 240: ;
[3X4] 241: ; Ermitteln, ob Drucker=BUSY
[P30] 242: ;
[UK1] 243: PR_STAT:
[V91] 244: PUSH DX
[F91] 245: PUSH AX
[R91] 246: PUSH CX
[LR1] 247: PR_GETST:
[2D4] 248: MOV DX,LPT2 ;Druckerport
[4K1] 249: MOV AH,02
[4V4] 250: INT INT_17 ;Printer-Status
[5P1] 251: PR_STAT1:
[P30] 252: ;
[M23] 253: ; Bit-Setting bei Return
[UD1] 254: ; 7 = busy
[541] 255: ; 6 = ack
[4S2] 256: ; 5 = out-of-paper
[E22] 257: ; 4 = selected
[LA2] 258: ; 3 = I/O-error
[M12] 259: ; 2/1 = unused
[V12] 260: ; 0 = time-out
[P30] 261: ;
[6E3] 262: AND AH,10000000B ;BUSY
[CG1] 263: CMP AH,00
[LZ3] 264: JZ PR_STAT2 ;0 = BUSY
[HX4] 265: JMP SHORT PR_RET ;nicht BUSY
[P30] 266: ;
[4X2] 267: ;printer is BUSY
[P30] 268: ;
[6P1] 269: PR_STAT2:
[6M1] 270: MOV CX,10
[7P1] 271: PR_STAT3:
[9P0] 272: NOP
[8A2] 273: LOOP PR_STAT3
[6G3] 274: JMP SHORT PR_GETST
[KE1] 275: PR_RET:
[PA1] 276: POP CX
[GAI] 277: POP AX
[TA1] 278: POP DX
[3P0] 279: RET
[VC2] 280: PR_STATP ENDP
[P30] 281: ;
[EL2] 282: BPRNT PROC NEAR
[P30] 283: ;
[YL5] 284: ; Druckbereich über BIOS-Druck auf
[ZS4] 285: ; Bildschirm drucken Eingabe:
[VH5] 286: ; SI = OFFSET Druckbereich Anfang
[755] 287: ; DX = OFFSET Druckbereich Ende
[ZU3] 288: ; BH = PAGE / SCREEN NR.
[N34] 289: ; CURP2 = Cursor Position
[P30] 290: ;
[9B1] 291: BPRNTE:
[P30] 292: ;
[041] 293: ;ENTRY
[P30] 294: ;
[C24] 295: MOV CX,01H ;REPETITION
[UR4] 296: MOV AH,0AH ;Funktion PRINT
[8R0] 297: PRF:
[VV6] 298: MOV AL,BYTE PTR CS:[SI] ;Zeichen holen
[LA3] 299: INT 010H ;drucken
[4Z4] 300: INC SI ;nächstes Zeichen
[PC1] 301: PUSH AX
[ZC1] 302: PUSH DX

```

Listing. »msprn.sys« – ein Druckertreiber im Selbstbau


```

[GV5] 303: MOV AH,02H ;SET-CURSOR-Funktion
[S82] 304: MOV DX,CURP2
[FW4] 305: ADD DL,01H ;nächste Spalte
[842] 306: MOV CURP2,DX
[XX3] 307: INT 010H ;SET-CURSOR
[TA1] 308: POP DX
[GA1] 309: POP AX
[XM3] 310: CMP DX,SI ;fertig?
[8N3] 311: JA PRF ;noch nicht
[3P0] 312: RET
[3P0] 313: ;
[G23] 314: ; Druck ist fertig
[3P0] 315: ;
[CR1] 316: BPRNT ENDP
[3P0] 317: ;
[R83] 318: PRNT_OUT PROC NEAR
[3P0] 319: ;
[SX3] 320: ; Zeichen aus AL drucken,
[ES3] 321: ; Drucker-Port ist in DX
[3P0] 322: ;
[LP4] 323: MOV AH,00H ;Funktions-Code
[0Y1] 324: INT INT_17
[3P0] 325: RET
[0D2] 326: PRNT_OUT ENDP
[3P0] 327: ;
[4P0] 328: PAGE
[3P0] 329: ;
[XA6] 330: ; ENTRY-Routine. Jeder INTERRUPT-CALL
[UN6] 331: ; kommt hier an und wird über COMM_TBL
[SG7] 332: ; gemäß dem Funktionscode auf das für ihn
[YG4] 333: ; bestimmte Label geleitet.
[3P0] 334: ;
[SE1] 335: MS_INT:
[PC1] 336: PUSH AX
[TC1] 337: PUSH BX
[WC1] 338: PUSH CX
[ZC1] 339: PUSH DX
[GC1] 340: PUSH SI
[3B1] 341: PUSH DI
[TB1] 342: PUSH BP
[BC1] 343: PUSH DS
[EC1] 344: PUSH ES
[6P7] 345: LDS BX,CS:[R_PAK] ;DS:BX Adr zum Requ.-H
eader
[3P0] 346: ;
[5Z5] 347: ; Überprüfen des Funktionscodes,
[PB3] 348: ; darf nicht > 16 sein.
[3P0] 349: ;
[488] 350: MOV AL,BYTE PTR DS:[BX].R_CODE ;Funktion
scode
[8N0] 351: CBW
[8C1] 352: PUSH CS
[5A1] 353: POP DS
[GH1] 354: CMP AL,16
[AS5] 355: JG CMD_ERR ;Funktionscode falsch
[PS2] 356: LEA SI,COMM_TBL
[HM1] 357: ADD SI,AX
[R85] 358: ADD SI,AX ;zwei mal, Doppelwort
[BD8] 359: JMP WORD PTR DS:[SI] ;Verzweigen über CO
MM_TBL
[YE1] 360: CMD_ERR:
[TV1] 361: MOV AH,09H
[K93] 362: LEA DX,MSG_6 ;MSG 6
[UX3] 363: INT 021H ;PRINT MSG_6
[752] 364: JMP MS_ERR2
[3P0] 365: ;
[4P0] 366: PAGE
[3P0] 367: ;
[FK6] 368: ; Zeichen aus der Transfer Area drucken.
[787] 369: ; WRITE WITH VERIFY wird ebenso behandelt.
[3P0] 370: ;
[YP1] 371: MS_PRINT:
[AP5] 372: LES BX,[R_PAK] ;Request Header
[BB7] 373: MOV DS,WORD PTR ES:[BX+16] ;Transfer-Are
a
[3V5] 374: MOV SI,WORD PTR ES:[BX].R_TRANS
[YB1] 375: MS_PRI:
[YV6] 376: MOV AL,BYTE PTR DS:[SI] ;Zeichen holen
[8C1] 377: PUSH CS
[5A1] 378: POP DS
[8P5] 379: CALL PR_STAT ;printer status
[YH1] 380: MOV AH,00
[KS1] 381: MOV SI,AX
[AN6] 382: MOV AL,PR_TAB [SI] ;Zeichen umwandeln
[Z15] 383: MOV DX,LPT2 ;PRINTERPORT
[FE6] 384: CALL PRNT_OUT ;Zeichen zum Drucker
[MX2] 385: LES BX,[R_PAK]
[NV1] 386: MOV CX,0001
[C18] 387: MOV ES:[BX].R_COUNT,CX ;no. of Bytes pri
nted
[4K6] 388: MOV WORD PTR ES:[BX].R_STAT,0100H ;DONE

```

```

[LK2] 389: JMP MS_RETURN
[3P0] 390: ;
[4P0] 391: PAGE
[3P0] 392: ;
[D46] 393: ; Drucker Status erfragen, wenn BUSY,
[CM6] 394: ; dann BUSY-Bit im Request Header setzen
[3P0] 395: ;
[ND1] 396: MS_STA:
[4K1] 397: MOV AH,02
[R55] 398: MOV DX,LPT2 ;LOGISCHER PRINTER
[6D5] 399: INT INT_17 ;GET PRINTER STATUS
[HX2] 400: LDS BX,[R_PAK]
[VD4] 401: TEST AH,10000000B ;BUSY-Bit ?
[MS3] 402: JZ MS_STA1 ;is BUSY
[S67] 403: MOV WORD PTR DS:[BX].R_STAT,0100H ;DONE-
Bit
[LK2] 404: JMP MS_RETURN
[3P0] 405: ;
[BE1] 406: MS_STA1:
[3P0] 407: ;
[XT5] 408: ; BUSY-Bit im Request Header setzen
[3P0] 409: ;
[K87] 410: MOV WORD PTR DS:[BX].R_STAT,0200H ;BUSY-
Bit
[LK2] 411: JMP MS_RETURN
[3P0] 412: ;
[4P0] 413: PAGE
[3P0] 414: ;
[3M5] 415: ; Fehlerausgang für alle Fehler
[3P0] 416: ;
[CF1] 417: MS_ERR1:
[MH7] 418: ; Unbekannter oder unerlaubter Funktionscod
e
[3P0] 419: ;
[C42] 420: LEA SI,MSG_5
[AB2] 421: LEA DX,MSG_5E
[3T1] 422: XOR BX,BX
[3P3] 423: CALL BPRNTE ;MSG_5
[GF1] 424: MS_ERR2:
[HX2] 425: LDS BX,[R_PAK]
[317] 426: MOV WORD PTR DS:[BX].R_STAT,100000000000
0011B
[8S4] 427: ; ERROR-, UNKNOWN-COMMAND-Bit
[LK2] 428: JMP MS_RETURN
[3P0] 429: ;
[LX1] 430: MS_OFUSH:
[3P0] 431: ;
[PN6] 432: ; OUTPUT FLUSH, sollte nicht vorkommen.
[YM6] 433: ; Wenn doch, wird Drucker initialisiert
[3P0] 434: ;
[1V1] 435: MOV AH,01H
[152] 436: MOV DX,LPT2
[0Y1] 437: INT INT_17
[HX2] 438: LDS BX,[R_PAK]
[S67] 439: MOV WORD PTR DS:[BX].R_STAT,0100H ;DONE-
Bit
[LK2] 440: JMP MS_RETURN
[3P0] 441: ;
[447] 442: SCHLUSS LABEL WORD ;Ende des Laufzeitcodes
[3P0] 443: ;
[4P0] 444: PAGE
[3P0] 445: ;
[TK1] 446: MS_INIT:
[3P0] 447: ;
[1D4] 448: ; INITIALIZATION-ROUTINE.
[S15] 449: ; Initialisieren des Druckers,
[W95] 450: ; Lesen CONFIG.SYS, BREAK-ADDRESS
[XU2] 451: ; setzen, RETURN.
[3P0] 452: ;
[7W2] 453: MOV BH,00 ;PAGE-#
[F02] 454: CALL GETCUR
[UX4] 455: ADD DH,02H ;DH=LINE, DL=COLUMN
[RV1] 456: MOV DL,00H
[954] 457: CALL SET_CUR ;SET CURSOR
[442] 458: LEA SI,MSG_1
[2B2] 459: LEA DX,MSG_1E
[3T1] 460: XOR BX,BX
[902] 461: CALL BPRNTE
[3P0] 462: ;
[8R3] 463: ; BEGIN OF CONFIG-CHECK
[3U4] 464: ; Parameter aus CONFIG-Zeile
[3T4] 465: ; lesen und Variable setzen.
[3P0] 466: ;
[HC1] 467: F_ENTR:
[HX2] 468: LDS BX,[R_PAK]
[8W4] 469: MOV AX,WORD PTR DS:[BX+20]
[L35] 470: ; SEGMENT zur CONFIG.SYS-Zeile
[BV4] 471: MOV DI,WORD PTR DS:[BX+18]
[415] 472: ; OFFSET zur CONFIG.SYS-Zeile
[GS1] 473: MOV ES,AX
[8C1] 474: PUSH CS

```



```

[5A1] 475: POP DS
[7D1] 476: F_ENTR1:
[S61] 477: INC DI
[C74] 478: MOV AL, BYTE PTR ES:[DI]
[5M1] 479: CMP AL, "/"
[812] 480: JZ F_ENTR3
[L04] 481: CMP AL, 0DH ;Zeilenende
[712] 482: JZ F_ENTR2
[A04] 483: CMP AL, 0AH ;Zeilenende
[712] 484: JZ F_ENTR2
[T13] 485: JMP SHORT F_ENTR1
[BD1] 486: F_ENTR2:
[P30] 487: ;
[UV5] 488: ; Keine Optionen angegeben, Fehler,
[U65] 489: ; der Driver wird nicht geladen
[P30] 490: ;
[1U1] 491: JMP F_RET
[FD1] 492: F_ENTR3:
[P30] 493: ;
[7N4] 494: ; "/" in Aufrufzeile gefunden,
[HA5] 495: ; Optionen suchen und in Variable
[R02] 496: ; eintragen.
[P30] 497: ;
[S61] 498: INC DI
[C74] 499: MOV AL, BYTE PTR ES:[DI]
[S24] 500: CMP AL, " " ;keine Leerstellen
[812] 501: JZ F_ENTR3
[8M1] 502: CMP AL, "0"
[EA2] 503: JNZ F_ENTR31
[YH1] 504: MOV AH, 00
[N96] 505: MOV BYTE PTR LPT2+1, AH ;Port Adresse 0
[523] 506: JMP SHORT F_ENTR5
[UG1] 507: F_ENTR31:
[BM1] 508: CMP AL, "1"
[GA2] 509: JNZ F_ENTR32
[1K1] 510: MOV AH, 01
[N96] 511: MOV BYTE PTR LPT2+1, AH ;Port Adresse 1
[523] 512: JMP SHORT F_ENTR5
[VG1] 513: F_ENTR32:
[EM1] 514: CMP AL, "2"
[KA2] 515: JNZ F_ENTR33
[4K1] 516: MOV AH, 02
[X96] 517: MOV BYTE PTR LPT2+1, AH ;Port Adresse 2
[523] 518: JMP SHORT F_ENTR5
[WG1] 519: F_ENTR33:
[P30] 520: ;
[RX4] 521: ; Es ist eine nicht zulässige
[195] 522: ; Port-Adresse angegeben worden,
[U65] 523: ; der Driver wird nicht geladen
[P30] 524: ;
[TN1] 525: JMP F_RET
[P30] 526: ;
[RD1] 527: F_ENTR5:
[0E5] 528: ; Adresse für logischen Drucker
[A95] 529: ; LPT2 ist gesetzt, jetzt Adresse
[B43] 530: ; für INT 17 setzen.
[P30] 531: ;
[S61] 532: INC DI
[C74] 533: MOV AL, BYTE PTR ES:[DI]
[SK1] 534: CMP AL, " "
[UA5] 535: JZ F_ENTR5 ;keine Leerstellen
[915] 536: CMP AL, "," ;Komma überlesen
[A12] 537: JZ F_ENTR5
[8M1] 538: CMP AL, "0"
[F72] 539: JNZ F_ENTR6
[YH1] 540: MOV AH, 00
[ZE6] 541: MOV BYTE PTR IPORT+1, AH ;Port Adresse 0
[S16] 542: JMP SHORT F_ENTR10 ;INT 17 Adresse
[VD1] 543: F_ENTR6:
[BM1] 544: CMP AL, "1"
[G72] 545: JNZ F_ENTR7
[1K1] 546: MOV AH, 01
[0F6] 547: MOV BYTE PTR IPORT+1, AH ;Port Adresse 1
[S16] 548: JMP SHORT F_ENTR10 ;INT 17 Adresse
[ZD1] 549: F_ENTR7:
[EM1] 550: CMP AL, "2"
[H72] 551: JNZ F_ENTR8
[4K1] 552: MOV AH, 02
[1F6] 553: MOV BYTE PTR IPORT+1, AH ;Port Adresse 2
[S16] 554: JMP SHORT F_ENTR10 ;INT 17 Adresse
[3E1] 555: F_ENTR8:
[N47] 556: ; Es ist eine nicht zulässige Port-Adresse
[3C7] 557: ; angegeben worden, Driver wird nicht gelad
en
[P30] 558: ;
[T21] 559: JMP F_RET
[P30] 560: ;
[HG1] 561: F_ENTR10:
[P30] 562: ;
[5D7] 563: ; Die Port-Adressen sind gesetzt, der Druck
er

```

```

[AN5] 564: ; kann jetzt initialisiert werden.
[P30] 565: ;
[1K1] 566: MOV AH, 01
[XC5] 567: MOV DX, LPT2 ;logischer Drucker
[7A1] 568: INT 017H
[P30] 569: ;
[LA7] 570: ; INT 17-CONTROLLER muss aufgesetzt werden
[P30] 571: ;
[YS1] 572: XOR AX, AX
[DS1] 573: MOV DS, AX
[3Y5] 574: MOV AX, DS:[017H*4+2] ;SEGMENT, INT 17
[6L5] 575: MOV BX, DS:[017H*4] ;OFFSET, INT 17
[EW3] 576: MOV DS:[INT_17*4+2], AX
[6H3] 577: MOV DS:[INT_17*4], BX
[8C1] 578: PUSH CS
[5A1] 579: POP DS
[R13] 580: LEA BX, MS_PRINTX
[YS1] 581: XOR AX, AX
[DS1] 582: MOV DS, AX
[DS1] 583: MOV AX, CS
[R73] 584: MOV DS:[017H*4+2], AX
[B43] 585: MOV DS:[017H*4], BX
[8C1] 586: PUSH CS
[5A1] 587: POP DS
[FT2] 588: MOV BH, 00 ;PAGE-#
[F02] 589: CALL GETCUR
[025] 590: ADD DH, 01H ;DH=LINE, DL=COLUMN
[RV1] 591: MOV DL, 00H
[954] 592: CALL SET_CUR ;SET CURSOR
[642] 593: LEA SI, MSG_2
[4B2] 594: LEA DX, MSG_2E
[3T1] 595: XOR BX, BX
[902] 596: CALL BPRNTE
[P30] 597: ;
[FM6] 598: ; INT 17-Handler ist aufgesetzt, RETURN
[P30] 599: ;
[3C2] 600: JMP F_RETURN
[P30] 601: ;
[C41] 602: F_RET:
[P30] 603: ;
[V95] 604: ; Status, End Adresse setzen, der
[1M4] 605: ; Driver wird nicht geladen
[P30] 606: ;
[HX2] 607: LDS BX, [R_PAK]
[A95] 608: MOV BYTE PTR DS:[BX].R_MEDIA, 00
[5M5] 609: MOV WORD PTR DS:[BX].R_TRANS, 0002
[5H4] 610: MOV WORD PTR DS:[BX+16], CS
[Y57] 611: MOV WORD PTR DS:[BX].R_STAT, 0100H ;DONE
Bit
[8C1] 612: PUSH CS
[5A1] 613: POP DS
[3M2] 614: MOV BH, 00 ;PAGE-#
[F02] 615: CALL GETCUR
[025] 616: ADD DH, 01H ;DH=LINE, DL=COLUMN
[RV1] 617: MOV DL, 00H
[954] 618: CALL SET_CUR ;SET CURSOR
[842] 619: LEA SI, MSG_3
[6B2] 620: LEA DX, MSG_3E
[3T1] 621: XOR BX, BX
[902] 622: CALL BPRNTE
[LK2] 623: JMP MS_RETURN
[P30] 624: ;
[7R1] 625: F_RETURN:
[P30] 626: ;
[CS7] 627: ; Status, End-Adresse setzen und Return zu D
OS
[P30] 628: ;
[M42] 629: ; SET STATUS
[P30] 630: ;
[HX2] 631: LDS BX, [R_PAK]
[Y57] 632: MOV WORD PTR DS:[BX].R_STAT, 0100H ;DONE
Bit
[P30] 633: ;
[D83] 634: ; End Adresse setzen
[P30] 635: ;
[4R2] 636: LEA AX, SCHLUSS
[L47] 637: MOV DS:[BX].R_TRANS, AX ;TRANSF-AREA (+14
)
[AA5] 638: MOV DS:[BX+16], CS ;TRANSF-AREA
[P30] 639: ;
[LK2] 640: JMP MS_RETURN
[P30] 641: ;
[632] 642: MS_PRN ENDS
[P30] 643: ;
[4M0] 644: END

```

Listing. »msprn.sys« – ein Druckertreiber im Selbstbau

Impressum

Herausgeber

Christian Widuch

Chefredaktion

Andreas Hagedorn (hg), Volker Everts (ev)

Stellvertretender Chefredakteur

Martin Aschoff (ma)

Redaktion

Dieter Bäumler (db), Gabi Börger (gb), Ulli Eike (ue),
Johann Glas (gh), Eva-Maria Hierlmeier (hi),
Dieter Müller (dm), Thomas Röder (tr), Andrea Scheduling (as),
Stefan Willmeroth (sw), Johann Zieger (zi)

Schlußredaktion

Eva-Maria Hierlmeier, Markus Stein, Andrea Voß

Redaktionsassistenten

Dorothea Lehner

Koordination

Stefan Ritter

Produktionsleitung

Gerd Köberich, Helmut Skoupy

Layout

Yvonne Hendricks, Patricia Reifenhäusen

Satz

Claudia Küllmer, Silvia Führer, Martina Siebert,
Gabriela Joseph, Marcus Geppert

Lektorat

Susanne Mias

Reprografie und Gestaltung

Manuela Eska, Margarete Schenk, Dieter Schnobl,
Silvia Erbrich, Andrea Gundlach

Werbegestaltung

Mohamed Hawa, Petra Biehl

Fotografie

Christian Heckmann, Klaus Jatho

Anzeigenleitung

Wolfgang Schnell

Es gilt die Anzeigenpreisliste Nr. 2 vom 01.06. 1988

Anzeigenverkauf

Wolfgang Brill, Gerlinde Rachow

Anschrift Redaktion:

Redaktionsbüro Everts & Hagedorn
Hans-Stießberger-Straße 3, 8013 Haar/München
Tel. (089) 46 10 01-0, Telefax (089) 46 10 01 90

Anschrift Verlag:

DMV - Daten & Medien Verlagsges. mbH
Postfach 250, Fuldaer Straße 6
3440 Eschwege

Telefon: (0 56 51) 80 09-0

Telex 993 210 dmv d

Telefax 0 56 51 / 80 09 33

Verlagsbüro München

Hans-Stießberger-Str. 3

8013 Haar/München

Leiterin Britta Fiebig

Vertrieb

Verlagsunion

Friedrich-Bergius-Straße 20

6200 Wiesbaden

Druck

Druckerei Jungfer, 3420 Herzberg

Bezugspreise

DOS EXTRA erscheint viermal jährlich,

Einzelpreis 18,- DM, 145 Ös, 18,- sfr.

Bankverbindungen:

Postscheck Frankfurt/M: Kto.-Nr.: 23043-608

Raiffeisenbank Eschwege:

BLZ: 522 603 85, Kto.-Nr.: 245 7008

Manuskripte, Programme:

bitte an das Redaktionsbüro Everts & Hagedorn,

Redaktion DOS International,

Hans-Stießberger-Str. 3, 8013 Haar/München senden.

Für unverlangt eingesandte Manuskripte und Datenträger sowie Fotos übernehmen der Verlag und die Redaktion keine Haftung. Die Zustimmung zum Abdruck wird vorausgesetzt.

Das Urheberrecht für veröffentlichte Manuskripte liegt ausschließlich beim Verlag. Nachdruck sowie Vervielfältigung oder sonstige Verwertung von Texten nur mit schriftlicher Genehmigung des Verlages. Für veröffentlichte Programme und Basteleien übernehmen wir weder Gewähr noch Haftung. Schaltungen und verwendete Zeichnungen müssen nicht frei von gewerblichen Schutzrechten sein.

Namentlich gekennzeichnete Fremdbeiträge geben nicht in jedem Fall die Meinung der Redaktion wieder.

Die Inserenten

AB Computersysteme.....	61
Albatros.....	46
Ing. Büro Bitter.....	21
Computer Solution.....	2
CSoft.....	117
CWTG.....	111
Data Becker.....	114,115
DMV-Verlag.....	9,11,17,37,51,121,125
Ing. Büro Fricke.....	66
Gude.....	61
Haase + Menrad.....	29
Imperial Software-Systems.....	77
Interest-Verlag.....	107
Janus-Elektronik.....	111
Kayser-Datentechnik.....	71
Kessler Software.....	66
Kieckbusch.....	53
Kirschbaum Software.....	25
Mental Connection.....	171
PC Colleg.....	46
R + P GmbH.....	57
TeWi-Verlag.....	172
Zoschke.....	111

Wer — wo — wann?

Bei Zuschriften bitten wir, folgendes zu beachten: Haben Sie Fragen, Anregungen oder Programme, die Sie der DOS-Redaktion zukommen lassen möchten, adressieren Sie wie folgt:

Redaktion

DOS International

Hans-Stießberger-Str. 3

8013 Haar/München

Tel. (089) 46 10 01-0

Haben Sie Fragen, die zum Beispiel Abonnements, Kleinanzeigen oder Databox-Bestellungen betreffen, wenden Sie sich bitte an den Verlag:

DMV-Verlag GmbH

DOS International

Fuldaer Straße 6

3440 Eschwege

Tel. (0 56 51) 80 09-0

Telex 993 210 dmv d

Telefax (0 56 51) 80 09-33

UniText PC

Das »fabelhafte« Textsystem

UniText. Viele der Leistungsmerkmale von UniText gehören bei anderen Textverarbeitungsprogrammen in das Reich der Legenden und Sagen.

UniText. Intelligente Textverarbeitung. Moderne Datenfernübertragung. Neuartige Adreßverwaltung. Komfortabler Dateimanager.

UniText. Ein ausgereiftes Komplettpaket:

Textverarbeitung

- Textgröße nur durch Kapazität der Diskette/Festplatte begrenzt
- erweiterbare, automatische deutsche Rechtschreibkorrektur (30000 Worte)
- Serienbrieffunktion
- deutsche Silbentrennung
- im- und exportieren von ASCII- und WordStar-Dateien
- auch Sonderzeichen können benutzt werden
- Druckertreiber für alle gängigen Matrix-, Typenrad- und Laserdrucker

Adreßverwaltung

- Dateigröße nur durch Kapazität der Diskette/Festplatte begrenzt
- automatische Anredeerkennung
- Etikettendruck, Druck auf Einzelblatt- und Endlospapier
- suchen und sortieren nach allen Kriterien
- im- und exportieren von ASCII-Dateien

Datenfernübertragung

- 11 verschiedene Baudraten
- unterschiedliche Bildschirmemulationen (VT, ANSI, ASCII)
- senden und empfangen von Dateien
- umfangreiche Parametereinstellungen (Protokoll, Datenbits, Stopbits, Parität)
- automatische ASCII-Anpassung
- großer Empfangs- und Sendepuffer

Dateimanager

- löschen, kopieren und umbenennen von Dateien
- ansehen von Dateiinhalten
- grafische Darstellung der Verzeichnisstruktur

UniText paßt sich dem Benutzer an. Gesteuert wird es kinderleicht über Pulldown-Menüs mit der Maus oder den Cursortasten oder über Control-Tasten.

UniText. In allen Programmteilen sind Schreibtischhilfen wie Taschenrechner, Notizblock, Uhr und Hilfsbildschirme aufrufbar.

UniText. Bücher schreiben und Mailboxen anrufen, Serienbriefe erstellen und Datenbanken benutzen, Festplatte organisieren und Kundendateien erstellen, Diktate korrigieren und Bewerbungen verschicken, Diplomarbeit schreiben und Vereinsmitglieder verwalten, Zeitungen machen und, und, und ...

UniText. Das "fabel"hafte Textsystem. Mit ausführlicher deutscher Bedienungsanleitung.

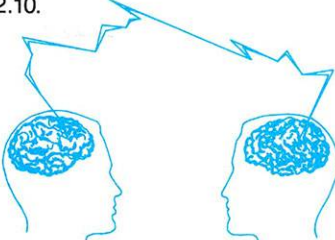
UniText. Für Hausfrauen, Manager, Journalisten, Studenten, Sekretärinnen, Beamte, Rechtsanwälte, Ärzte, Lehrer, Übersetzer, Ingenieure, Geistliche, Spediteure, Selbständige, Einzelhändler ...

UniText. Lauffähig auf allen gebräuchlichen Grafikkarten. Für jeden IBM-kompatiblen PC, XT oder AT mit mindestens 256 KByte Hauptspeicher, mindestens einem Diskettenlaufwerk und MS-DOS oder PC-DOS ab Version 2.10.

Händleranfragen erwünscht

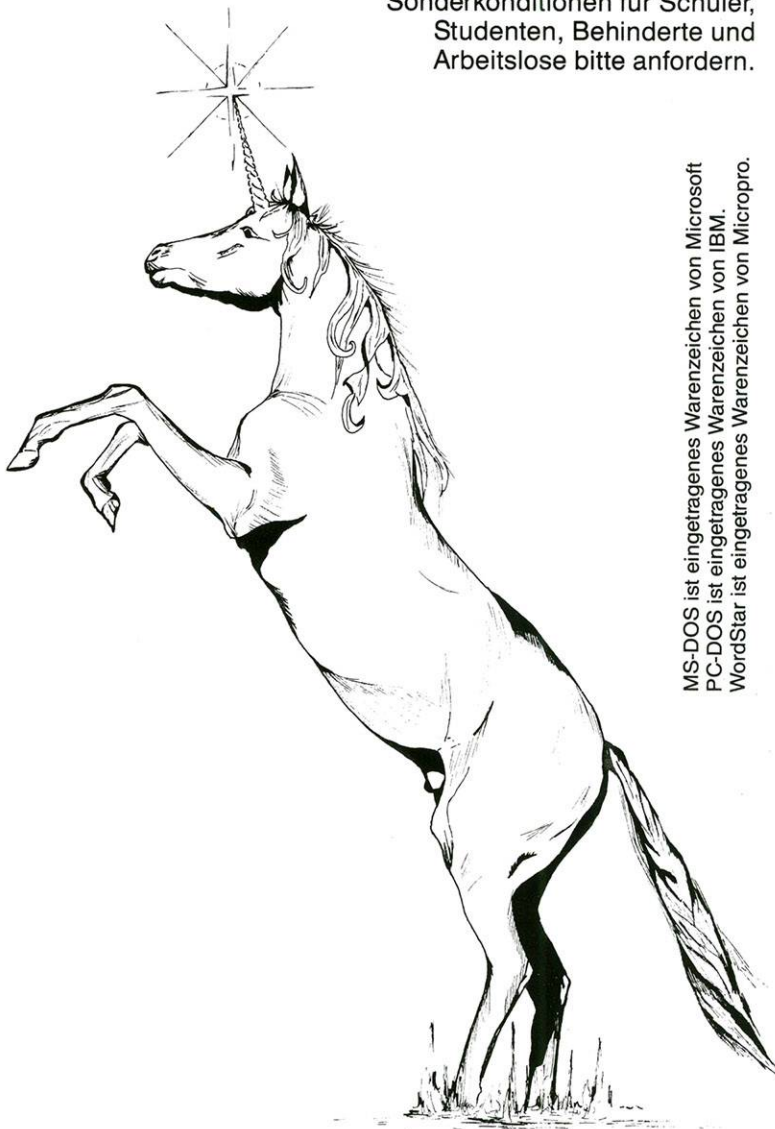
mental connection

Markus Zietlow
Schwelmer Straße 119
D-5820 Gevelsberg
Telefon: 0 23 32/5 14 43



Das Unternehmen für innovative Computertechnik

Sonderkonditionen für Schüler,
Studenten, Behinderte und
Arbeitslose bitte anfordern.



MS-DOS ist eingetragenes Warenzeichen von Microsoft
PC-DOS ist eingetragenes Warenzeichen von IBM.
WordStar ist eingetragenes Warenzeichen von Micropro.

Für märchenhafte

99, — DM

unverbindliche Preisempfehlung

Bitte Coupon ausschneiden und an nebenstehende Adresse schicken

- ☐ Bitte senden Sie mir ____ Exemplare von UniText PC DOS E 5/88
☐ per Nachnahme ☐ per Vorausscheck
☐ Ich benötige nähere Informationen zu UniText PC
☐ Bitte senden Sie mir eine Demo-Diskette (10,- DM in bar anbei)

Name _____

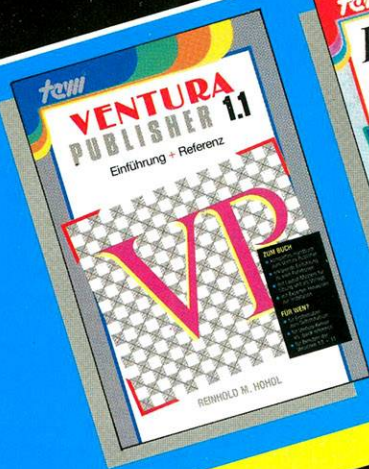
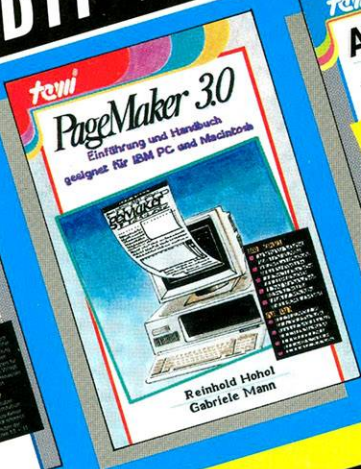
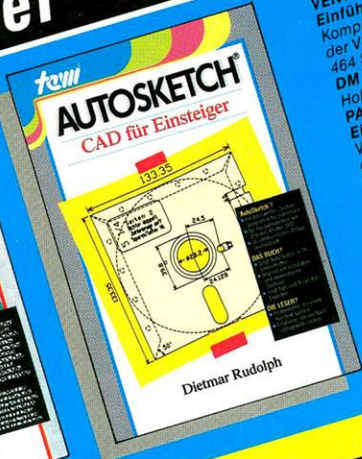
Straße _____

Ort _____

Datum, Unterschrift _____

Werkzeuge heutiger DESKTOP-PUBLISHER...

Die DTP-Macher

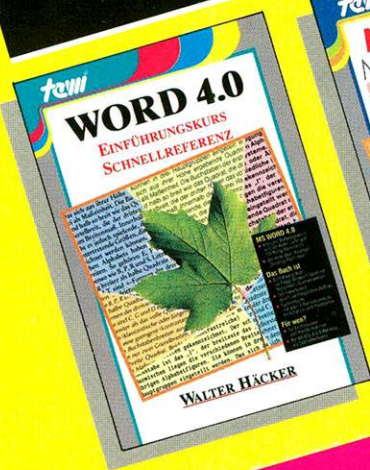
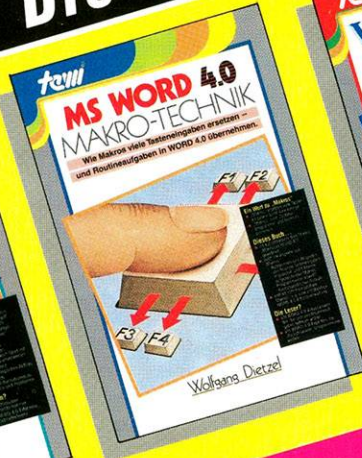


R. Hohol
VENTURA PUBLISHER 1.1
Einführung + Referenz
Komplettes Handbuch mit Dokumentation
der VENTURA-Anwendung.
464 Seiten, Hardcover,
DM 79,-/sFr 72,20/6S 616,20

Hohol/Mann
PAGEMAKER 3.0
Einführung + Handbuch
Vollständiges Handbuch mit Anwendun-
gen. Für IBM-Benutzer. Zeigt alle Page-
maker-Funktionen auch für Macintosh.
450 Seiten, Hardcover,
DM 79,-/sFr 72,20/6S 616,20

D. Rudolph
AUTOSKETCH
CAD für Einsteiger
In 26 Lernschritten von der Installation
über Zeichenübungen bis zur Zeich-
nungsübernahme in VENTURA.
272 Seiten, Hardcover,
DM 59,-/sFr 54,30/6S 460,20

Die DTP-Helfer



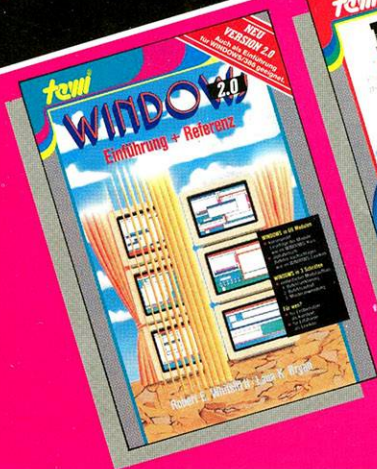
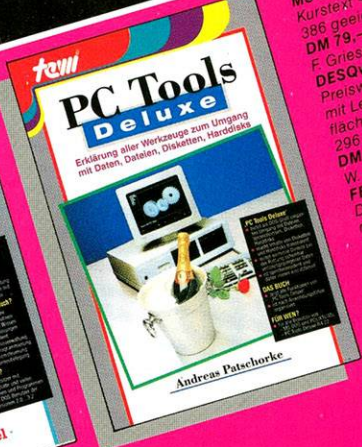
W. Hacker
WORD 4.0
Einführungskurs + Schnellreferenz
350 Seiten, Hardcover,
DM 69,-/sFr 63,50/6S 538,20

W. Dietzel
MS WORD 4.0 Makro-Technik
Wie ersetzt man Alltagsroutinen durch
Makros? Zeigt sofort einsetzbare Makro-
200 Seiten, Hardcover,
DM 59,-/sFr 54,30/6S 460,20

J. Gold
WordPerfect 4.2
Einführung + Referenz.
Best-rezensierter Kurstext.
496 Seiten, Hardcover,
DM 69,-/sFr 63,50/6S 538,20

Ph. Luidl
Desktop Knigge
Stil und Normen des Druck-
gewerbes für Desktop-Publisher.
200 Seiten, Hardcover,
DM 79,-/sFr 72,20/6S 616,20

Das DTP-Fundament



Whitsitt/Bryan
MS WINDOWS 2.0: Einführung + Referenz
Kurstext + Lexikon. Auch für WINDOWS!
386 geeignet. 496 Seiten, Hardcover,
DM 79,-/sFr 72,20/6S 616,20

F. Grieser
DESQview
Preiswerte Benutzeroberfläche für alle PCs
mit Leistungen künftiger 80386-Ober-
flächen. Eine WINDOWS-Alternative.
296 Seiten, Hardcover,
DM 59,-/sFr 54,30/6S 460,20

W. Dietzel
FESTPLATTENVERWALTUNG
Daten und Programme professionell
mit MS DOS-Befehlen verwalten.
190 Seiten, Softcover,
DM 39,-/sFr 35,90/6S 304,20

A. Patschorke
PC Tools Deluxe
Nach Anwendungsfallen organi-
siert. Eine NORTON-Alternative.
160 Seiten, Hardcover,
DM 49,-/sFr 45,10/6S 382,20



te-wi Verlag GmbH
Theo-Prosel-Weg 1
8000 München 40

... vom PC-FACHBUCHVERLAG „te-wi“